KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Markus Sorjonen

OHJELMOINNIN OPISKELUN PEDAGOGIIKKA SEKÄ OHJELMOINTIYMPÄRISTÖN RAKENTAMINEN

Opinnäytetyö
Huhtikuu 2020

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Huhtikuu 2020
Tietojenkäsittelyn koulutus

Karjalankatu 3
80220 JOENSUU
(013) 260 600

Tekijä(t)
Markus Sorjonen

Nimeke
Ohjelmoinnin opiskelun pedagogiikka sekä ohjelmointiympäristön rakentaminen

Toimeksiantaja
Karelia-ammattikorkeakoulu

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli perehtyä ohjelmoinnin opiskeluun yleisesti ja siihen, millaisia haasteita opiskeluun liittyy. Tavoitteena oli myös löytää ratkaisuja löydettyihin haasteisiin sekä rakentaa ohjelmoinnin opiskelua tukeva opiskeluympäristö. Ohjelmointiympäristö pyrki tukemaan opiskelijan kehitystä tarjoamalla yhtenäistetyn menetelmän ohjelmointikurssien suorittamiseen.

Opinnäytetyössä tarkasteltiin ohjelmoinnin opiskelun lähtökohtia, käsiteltiin sen pedagogiikkaa sekä esiteltiin ohjelmoinnin opiskeluun tarkoitetun ympäristön toteutusta. Empiirisessä osiossa esiteltiin markkinoilla olevia ohjelmointiympäristöjä sekä kuvailtiin, millaisia ominaisuuksia palveluista löytyi.

Opinnäytetyön tuloksena huomattiin, että opiskelijan motivaatiolla voi olla useita eri vaikuttimia. Tarkastelimme myös erilaisia vaihtoehtoja opiskelijan motivaation säilyttämiseksi ohjelmointiopiskelussa. Toteutettavana työnä tehtiin toimiva ohjelmointiympäristö opiskelun tukemiseen olemassa olevista työkaluista ja palveluista, jotka toimivat toisistaan riippumatta.

Kieli
suomi

Sivuja 44

Asiasanat
Ohjelmointi, oppiminen, pedagogiikka

THESIS
March 2020
Business Information Technology

Karjalankatu 3
80220 JOENSUU
FINLAND
(013) 260 600

Author (s)
Markus Sorjonen

Title
The Pedagogy of Programming Study and Building Learning Environment

Commissioned by
Karelia University of Applied Sciences (UAS)

Abstract

The aim of this thesis was to become familiar with the study of programming in general and learn what the challenges involved in programming studies are. The purpose was to find solutions to challenges found and build a virtual environment that supports the study of programming. This environment aspired to support learning programming by providing a unified method for completing programming courses and tasks.

This thesis examined the studying of programming, its pedagogy, and the implementation of the programming environment. The empirical part introduced most popular current learning environments on the market and described the main features found in these services.

As a result of the thesis, it was discovered that student motivation has several different factors and there are different options for maintaining student motivation. This thesis also introduces a functional programming environment to support learning made with already existing tools and services.

Language

Finnish

Pages 44

Keywords

programming, learning, pedagogy

Sisältö

1 Johdanto

Ohjelmointityöskentelijöiden määrä on maailmanlaajuisesti kasvussa. Yhdysvalloissa ohjelmoijien määrä oli 1 256 200 henkilöä vuonna 2016, ja työpaikkojen määrän on ennustettu kasvavan 24 % kymmenen vuoden aikana (Bureau of Labor Statistics, U.S. Department of Labor 2018). Maailmanlaajuisesti ohjelmoijia on 18.2 miljoonaa vuonna 2017, ja sen on ennustettu kasvavan 26.4 miljoonaan vuoteen 2019 mennessä (Evans Data Corporation, 2017). Suomessa tietotekniikka-alan yrityksissä on noin 90 000 henkilöä vuonna 2015 ohjelmisto- ja palveluyrityksissä (Neittaanmäki & Kinnunen 2016).

Ohjelmointiopiskelu on Suomessa kasvussa. Vuonna 2014 joulukuussa opetushallitus tiedotti ohjelmoinnin siirtyvän perusopetuksen opetussuunnitelmaan, jonka perusteella ohjelmointia on aloitettu opettaa kaikilla vuosiluokilla vuoden 2016 syksystä lähtien (Opetushallitus 2014). Ohjelmointityöskentelijöiden tarve on maailmanlaajuisesti nousussa. Tämä näkyy osaavan työvoiman tarpeen lisääntymisenä sekä ohjelmointiopetuksen määrän lisääntymisenä. Talouselämän artikkelissa kerrotaan Tieto- ja viestintätekniikan TIVIA ry:n arvioineen Suomeen tarvittavan 9 000 uutta ohjelmistoammattilaista ja tarpeen kasvavan 3 800 henkilöllä vuodessa (Muukkonen 2017).

Ohjelmointityöskentelijöiden määrän kasvu on tuonut markkinoille useita ohjelmointiopiskeluun tarkoitettuja palveluita, joiden avulla voi tutustua eri ohjelmointikielten perusteisiin tai täydentää oppimaansa eri aihealueilta ohjattuna itseopiskeluna. Opiskelumateriaalien saatavuus ja tarjonta eivät rajoitu pelkästään ohjelmointiin, vaan tarjolla on myös tietotekniikan eri aihealueille, kuten tietoturvallisuuteen liittyviä palveluita, jotka tarjoavat käyttäjälle tietoa. Tässä opinnäytetyössä keskitytään ohjautuviin palveluihin, joissa järjestelmä ohjaa käyttäjää opiskelussa ja jotka ovat keskittyneet ohjelmointiopetukseen Karelia-ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman opiskelijoille.

Opinnäytetyön tavoitteena on selvittää, miten opiskelijaa voidaan motivoida suoriutumaan parhaiten hänelle määritellyistä tehtävistä. Tämän perusteella on

pyrkimys toteuttaa teknisenä toteutuksena opetusympäristö, joka helpottaa ohjelmointiopiskelua. Tekninen toteutus ei ota kantaa ohjelmointikurssien kontekstiin, vaan keskittyy tarjoamaan opiskeluympäristön, joka on käytettävissä erilaisten ohjelmointikurssien kanssa. Tekninen toteutus rajataan koskemaan tietojenkäsittelyn opiskelijoiden ohjelmointiopiskelua Karelia-ammattikorkeakoulussa.

2 Ohjelmoinnin opiskelun pedagogiikkaa

Ohjelmointiopiskelua voi toteuttaa verkossa. Maailmalla on useita avoimia verkkokursseja, joiden kautta ohjelmointia voi opiskella verkkoluentojen avulla tai seuraamalla kurssilla käytävää materiaalia itsenäisesti. Useilla ihmisillä on halu opetella ohjelmointia, ja esimerkiksi Helsingin yliopiston tietojenkäsittelytieteen laitoksen kursseja on voinut vuodesta 2012 suorittaa vapaasti verkossa käytävinä MOOC-kursseina suomen kielellä (Kosola 2015). Helsingin yliopisto tarjoaa Ohjelmoinnin MOOC–kurssin avulla mahdollisuuden opinto-oikeuteen tietojenkäsittelytieteen opintoihin Helsingin yliopistolla. Kurssi tutustuttaa opiskelijan nykyaikaisen ohjelmoinnin perusasioihin (Helsingin Yliopisto 2018). Kansainvälisesti Harvardin yliopisto tarjoaa tietojenkäsittelytieteen alkeiskurssia, CS50's Introduction to Computer Science ilmaiseksi Edx-palvelun kautta (Edx 2018). Kurssilla tutustutaan tietojenkäsittelytieteen ja ohjelmoinnin alkeisiin.

Karelia-ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma on mahdollista suorittaa etäopintona opetuksen tapahtuessa verkon välityksellä (Karelia-ammattikorkeakoulu 2018). Opetus tapahtuu verkkoluentoina, ja tehtävämateriaali jaetaan verkkopalvelun kautta, johon jokainen opiskelija kirjautuu henkilökohtaisilla käyttäjätunnuksilla. Opiskelija voi palvelussa seurata kurssien etenemistä, palauttaa tehtäviä, osallistua luentoihin ja osallistua keskusteluun muiden opiskelijoiden sekä ohjaajien kanssa. Ohjelmointiopiskelijan näkökulmasta ongelma ilmenee siinä, kun opiskelija asentaa työasemalle ohjelmia ja sovelluksia, joita tarvitaan kurssilla palautettavien tehtävien tekemiseen. Markkinoilla on useita erilaisia

käyttöjärjestelmiä, sovelluksia ja palveluita, joita voi käyttää ohjelmointitehtävien tekemiseen, jotka eivät välttämättä ole yhteensopivia toistensa kanssa. Tämä aiheuttaa kurssin järjestäjän ja kurssin osallistujan näkökulmasta haasteita, jotka kuluttavat resursseja ja vievät aikaa opiskelulta.

2.1 Ohjelmointiopiskelun haasteet

Ohjelmointiopiskeluun liittyy useita haasteita, jotka voivat aiheuttaa ongelmia ohjelmointikurssien suorittamisessa. Ongelmat voivat hidastaa kurssin suoritusta tai pahimmassa tapauksessa aiheuttaa kurssin hylkäämisen. Bergin ja Reilly toteavat tutkielmassaan opiskelijoiden ohjelmointikursseilla kokemien ongelmien johtavan korkeisiin tilastoihin kurssien hylkäyslukemissa.

Kinnusen (2009) tutkimuksessa tarkasteltiin syitä CS1-kurssin (computer science) hylkäämiselle. Kurssin 500–600 osallistujasta kurssin kesken jättäneitä oli 30–50 %. Kvalitatiivisen tutkimuksen perusteella tutkielmassa todettiin yleisimmiksi syiksi ajan ja motivaation puute. Nämä molemmat syyt kuitenkin koostuivat useasta tekijästä, joita olivat muun muassa kurssin vaativuus sekä opiskelijan yleiset ajankäytölliset ongelmat. Tutkijoiden kysyessä ryhmältä syitä kurssin keskeyttämiseen, yleisimmät syyt olivat:

- tehtävien tekeminen vei liian kauan aikaa, ja opiskelija päätti lopettaa kurssin kesken keskittyäkseen muihin kursseihin.
- opiskelija ei tiennyt kuinka tehdä kurssitehtäviä.
- opiskelija aloitti tehtävien tekemisen liian myöhään, ja ei saanut niitä tehtyä määräajan puitteissa. (Kinnunen 2009.)

Tutkijat kuitenkin toteavat, että tulosten perusteella ei voida yksinkertaistaa syitä kurssin kesken jättämiselle. Mahdollisia syitä voi olla useita, ja monesti syyt kumuloituvat opiskelijalle yksilöllisesti. Näin toimenpiteet opiskelijoiden motivoimiseen nähdään haastavana: joitakin voi auttaa visuaalinen oppiminen tai opiskeluympäristö, mutta se ei auta kaikkia (Kinnunen 2009). Teoreettisena haasteena on saada opiskelija kiinnostumaan aiheesta ja motivoida suoriutumaan kurssilla määritellyistä tehtävistä.

Haasteet voivat ilmetä myös itse ohjelmointityöskentelyyn liittyvinä teknisinä käytännön ongelmina. Eri sovellusten ja työkalujen välillä voi ilmetä yhteensopivuusongelmia, jos yhteensopivuutta ei ole erikseen huomioitu. Markkinoilla on saatavilla lukuisia erilaisia käyttöjärjestelmiä, teknisiä laitteita, ohjelmistoja ja palveluita, joiden yhteensopivuutta toisiinsa on mahdotonta ennakoida tarkasti etukäteen. Jos kurssin materiaali on suunniteltu käyttäen Windows-käyttöjärjestelmää, voi eri käyttöjärjestelmää käyttävä opiskelija ajautua ongelmiin tehtäviä suorittaessaan. Eri sovellukset toimivat eri käyttöjärjestelmissä, ja vaikka sovellus olisi tuettu monessa käyttöjärjestelmässä, niiden asentaminen ja käyttö voi olla erilaista. Näiden yhteensopivuusongelmien ratkaisemiseen menee resursseja kurssin järjestäjän selvittäessä opiskelijoiden kokemia ongelmia, ja siihen menee myös opiskelijalta aikaa, jonka voisi käyttää kurssimateriaalin opiskeluun.

2.2 Strategioita ongelmien ylittämiseen

Berglundin ja Eckerdalin (2015) mukaan opiskelijoita täytyy rohkaista oppimaan ohjelmointia samanaikaisesti, teoreettisesta sekä käytännöllisestä näkökulmasta. Heidän mielestään toisen osa-alueen laiminlyöminen johtaa heikkoon oppimiseen tai mahdollisesti epäonnistumiseen. Testiryhmästä saatujen tulosten perusteella tutkijat huomasivat, että vasta opiskelijoiden tuottaessa omaa koodia he pystyivät vertailemaan eri esimerkkejä sekä huomaamaan, mitä heidän täytyy oppia, jotta he voivat saavuttaa halutun lopputuloksen. (Eckerdal & Berglund 2015.)

Berglundin ja Eckerdalin (2015) tutkimuksen testiryhmässä osallistujat etenivät tekstieditorin sekä dokumentaation avulla ohjelmointipulmaan, jossa oli ohjelmointikielen syntaksin teoreettisia ohjeita sekä esimerkkejä. Osallistujilla ei ollut merkittävää aiempaa ohjelmointikokemusta. Tutkimuksessa esiteltiin testiryhmän törmänneen seuraavanlaiseen ohjelmointitehtävään (kuvio 1).

```
public void distClosestWall() {
    this.getModelDisplay().getHeigth();
    this.getModelDisplay().getWidth();
    if (getHeight()<getWidth());
        System.out.println(int getHeiget());
    else
        System.out.println(int getWidth());
```

Kuvio 1. Kuvaus testiryhmän ohjelmointikoodin lähtötilanteesta (Eckerdal & Berglund 2015.)

Kyseinen koodi sisälsi syntaksivirheitä, ja osallistujien tehtävänä oli muokata koodia niin, että se on oikein. Tässä vaiheessa osallistujat eivät vielä tienneet, mikä puolipisteen merkitys koodissa on. Osallistujat avasivat dokumentaation, ja kohtasivat esimerkkejä ehtolauseista (kuvio 2). (Eckerdal & Berglund 2015.)

```
Esimerkki 1
if (x > 0) {
 System.out.println(“Positive.”);
}

Esimerkki 2
if (x > 0) {
 System.out.println(“Positive.”);
}
else {
 System.out.println(“Not positive.”);
}
```

Kuvio 2. Esimerkkejä ehtolauseista (Eckerdal & Berglund 2015.)

Tässä vaiheessa osallistujat huomasivat eron esimerkkien sekä oman koodinsa välillä. Opiskelijat kykenivät erottamaan koodissa puolipisteiden tai aaltosulkumerkkien sijainnin, sekä mahdollisesti havaitsemaan, mihin lisäykset tulisivat omassa koodissa tehdä. Osallistujat eivät kuitenkaan tienneet niiden tarkoitusta, ja missä tai miksi ehtolauseessa sulkumerkkejä tulee käyttää. Osallistujat keskustelivat ryhmässä tilanteesta ja kokeilivat sijoittaa aaltosulkumerkkejä vertailemalla niiden sijaintia esimerkkien avulla. Lopulta he onnistuivat lisäämään aaltosulkumerkit ehtolauseen yhteyteen. Tämän vaiheen jälkeen koodi näytti tältä:

```
public void distClosestWall() {
    this.getModelDisplay().getHeigth();
    this.getModelDisplay().getWidth();
    if (getHeight()<getWidth()); {
        System.out.println(int getHeiget ());
    }
    else {
        System.out.println(int getWidth());
    }
}
```

Kuvio 3. Testiryhmän ohjelmointikoodi dokumentaatioon tutustumisen jälkeen (Eckerdal & Berglund 2015.)

Vaikka osallistujilla ei ollut teoreettista tietämystä, mistä syystä aaltosulkumerkkejä käytetään ehtolauseiden yhteydessä, he olivat onnistuneet lisäämään omaan koodinsa vastaavanlaiset aaltosulkumerkit. Syntaksista heillä ei vielä ollut täydellistä käsitystä, kuten kuvassa neljännen rivillä oleva puolipiste osoittaa. He eivät tienneet edelleen aaltosulkumerkkien tai puolipisteiden merkitystä, mutta esimerkkitapaukset kiinnittivät heidän huomionsa niiden käyttöön koodissa (Eckerdal & Berglund 2015).

Tutkimuksessa pyrittiin selvittämään monimutkaista yhteyttä opiskelijoiden teorian ja käytännön oppimisen välillä. Testiryhmästä saatujen tulosten perusteella tutkijat huomasivat yhtäläisyyksiä: osallistujat eivät havainneet eroja ehtolauseissa oman ja esimerkkikoodien välillä ennen kuin he kokeilivat muodostaa niitä itse. Kokeessa osallistujat havaitsivat koodien eron ja tulivat tietoiseksi omasta syntaksin tietämättömyydestä. Huomatessaan eron he pyrkivät korjaamaan sen ja lisäävän tietämystään teoreettisessa näkökulmassa, jos käytäntö ei riitä ratkaisemaan ongelmaa. Kun teoreettinen selitys oli oppijan mielestä tyydyttävä, hän jatkoi käytännöllisen toimenpiteen parissa. Tämä johti ”aaltoon”, jossa oppija vaihteli teoreettisen ja käytännöllisen oppimisen välillä (Eckerdal & Berglund 2015).

2.3 Motivaatio osana opiskelua

Motivaatio tai sen puute vaikuttaa suuresti kurssin hylkäämiseen. Motivaation määritelmä on kuitenkin laaja. Esimerkiksi Kotomaisten kielten keskuksen sanakirja määrittelee motivaation olevan ”johonkin toimintaan johtavien motiivien kokonaisuus” (Kielitoimiston sanakirja 2018). Tämä määritelmä kuitenkin jättää täysin avoimeksi motiivien alkuperän. Motiivilla nähdään olevan suuri vaikutus menestykseen koulussa ja työpaikalla, ja motivaation syntyyn vaikuttavat henkilön sisäiset ja ulkoiset tekijät. Kun henkilö toimii saadakseen palkinnon, esimerkiksi arvosanan kurssin suorittamisesta, hän on ulkoisesti motivoitunut. Henkilö, joka innostuu tehtävästä, esimerkiksi oman mielenkiinnon ohjaamana, on sisäisesti motivoitunut (Salovaara 2004).

Opiskelussa sisäisesti ja ulkoisesti motivoituneet opiskelijat toimivat eri tavoin. Ulkoisesti motivoitunut opiskelija todennäköisesti ei tee tehtäviä, joita ei huomioida arvioinnissa, mutta sisäisesti motivoitunut saattaa perehtyä materiaaliin syvällisemmin tai kurssin aihepiirin ulkopuolelta (Jenkins 2001). Tähän perustuen kurssin toteutuksen tulisi palvella eri tavoin motivoituneiden opiskelijoiden tarpeita. Ohjelmointikurssin aikana tulisi antaa opiskelijalle palautetta vertailevasti, ja samalla mahdollistaa aihealuetta syvällisempi paneutuminen aiheeseen. Suoritusta korostava arviointi sekä opiskelijoiden kannustaminen ymmärtämään opittava asia on suotavaa motivaation kannalta (Salovaara 2018).

3 Ohjelmointiympäristön toteutus

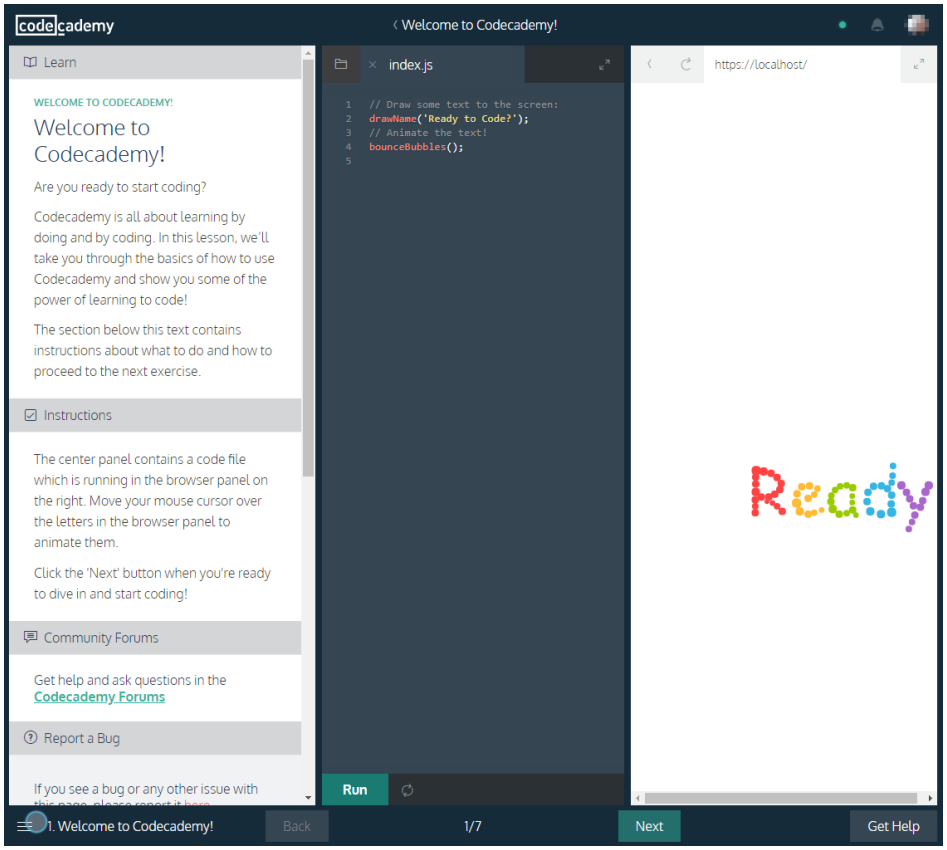
Ohjelmointiympäristön tarkoituksena on muodostaa käyttöympäristö ohjelmointikurssien suorittamiseen. Eri työkalujen tarve voi vaihdella riippuen kurssin tavoitteesta, mutta ohjelmointiympäristöön on valikoitu työkalut, jotka mahdollistavat tehtävien palautuksen, seurannan sekä kurssiarvostelun. Ympäristö ei lähde korvaamaan jo Karelia-ammattikorkeakoululla käytössä olevia portaaleja, vaan tarkoituksena on tukea erityisesti ohjelmointiopiskelun tavoitteita

kurssikohtaisesti näiden jo käytettävien portaalien rinnalla. Työssä käytetyt työkalut ja palvelut eivät ole käytön suhteen riippuvaisia toisistaan, joten kaikkia työkaluja ei ole pakollista sisällyttää kurssiin. Myös jokaisen käytetyn palvelun tai työkalun voi halutessaan vaihtaa toiseen, jos se koetaan hyödylliseksi kurssin tavoitteisiin nähden.

Ohjelmointiopiskelun kannalta on erityisen tärkeää, että käytettävät työkalut tukevat oppijaa parhaalla mahdollisella tavalla. Gomesin ja Mendesin (2007) mukaan on olemassa muutama olennainen tekijä, jotka ovat erityisen tärkeitä ohjelmointiopiskelun ympäristön suhteen:

- päivittyvä analyysi opiskelijan sen hetkisestä tasosta
- mukautuminen opiskelijan oppimistyylin mukaan
- toistuvien mallien käyttö opiskelijan tukena
- pelien käyttö (interaktiivisuus)
- mahdollisuus algoritmien suunnitteluun. (Gomes & Mendes 2007.)

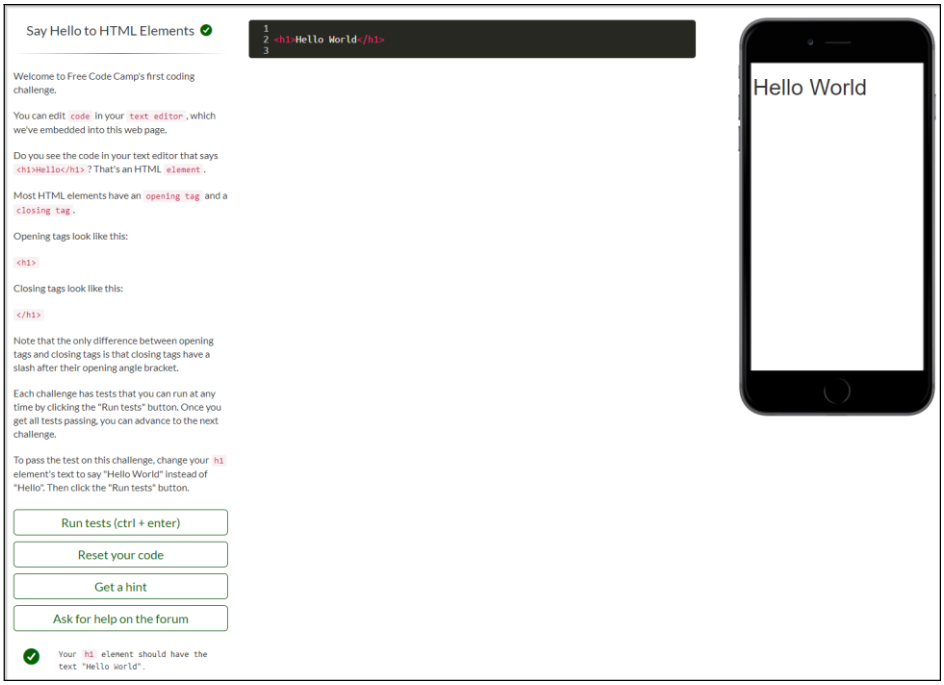
Monet vapaasti valittavissa olevat verkkokurssit noudattavat jollain tapaa näitä ohjeistuksia ohjelmoinnin opettamiseen. Codeacademy on yksi suosituimmista palveluista ohjelmoinnin opiskeluun verkossa (Falcon 2017). Palvelu tarjoaa interaktiivisen, ilmaisen ympäristön ja materiaalin web-ohjelmoinnin opetteluun. Se tarjoaa myös lisäominaisuuksia maksullisena lisäpalveluna. Palvelun opiskeluympäristössä on kolme osaa: materiaali (teoria), ohjelmointieditori (käytäntö) sekä selainnäkymä, josta ohjelmointikoodiin tehdyt muutokset päivittyvät automaattisesti (kuva 1).



Kuva 1. Codeacademyn opiskeluympäristö

Palvelun käyttäjä etenee vasemmalla olevan ohjeistuksen mukaan, jossa ohjeistaan käyttäjää muokkaamaan keskellä olevaa koodia ohjelmointieditorissa. Käyttäjän suorittaessa tehtävän ohjelma tunnistaa oikean vastauksen, näyttää oikealla selainnäkymässä käyttäjän muodostaman muutoksen sekä jatkaa eteenpäin antamalla uuden tehtävän (Codeacademy 2018).

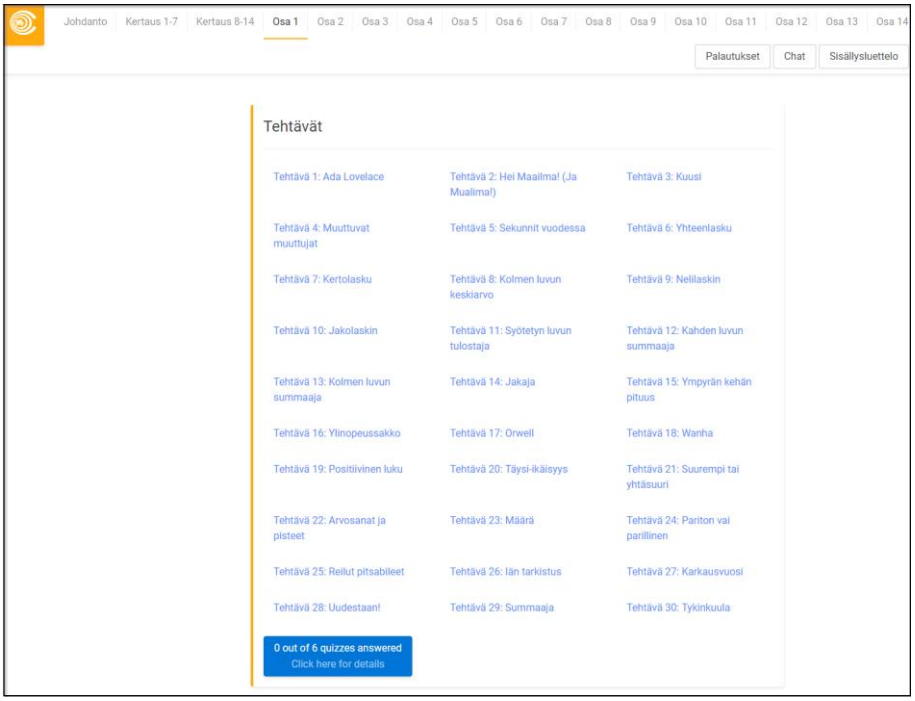
Moni vastaavanlainen palvelu tarjoaa samanlaisen ohjelmointiympäristön käyttäjilleen internetselaimen käyttöliittymällä. Toinen vapaaseen ohjelmointiopetukseen erikoistunut palvelu, freeCodeCamp, noudattaa samaa periaatetta palvelussaan kuin Codeacademy (kuva 2).



Kuva 2. freeCodeCampin ohjelmointiympäristö

Vasemmalla käyttäjä saa palvelun määrittelemän ohjelmointitehtävän, jonka käyttäjä ratkaisee käyttäen apuna keskellä olevaa ohjelmointieditoria, ja oikealla olevaa selainnäkymään päivittyy editoriin kirjoitettu ohjelmointikoodi. Ohjelma tunnistaa, kun käyttäjä on suorittanut tehtävän oikein, ja antaa uuden ohjelmointitehtävän. Palvelut näyttävät käyttäjälle sillä hetkellä ainoastaan niitä tietoja, jotka ovat oleellisia tehtävän suorittamisen kannalta. Molemmissa tapauksissa käyttäjän jäädessä jumiin tehtävää suorittaessaan hän voi avata vinkin, joka auttaa tehtävän ratkaisemisessa (freeCodeCamp 2018). Molemmissa palveluissa myös käsitellään useampaa eri aihetta, jotka liittyvät ohjelmointiin. Tällaisen palvelun avulla käyttäjälle voidaan esittää useita eri ideoita ja tekniikoita, joita ohjelmoinnissa käytetään. Palveluiden käyttö ei myöskään ole rajoitettu käyttäjämäärään, vaan niiden kautta yhä useammat henkilöt voivat perehtyä aiheisiin.

Helsingin Yliopiston MOOC-kurssi noudattaa perinteisempää virtuaalisen kurssimateriaalin rakennetta. Tehtävät ovat jaoteltu osa-alueisiin allekkain, ja jokainen osallistuja lataa työasemalla ohjelman, jonka avulla ohjelmointitehtävät suoritetaan (kuva 3).



Kuva 3. Helsingin Yliopiston MOOC-kurssin näkymä

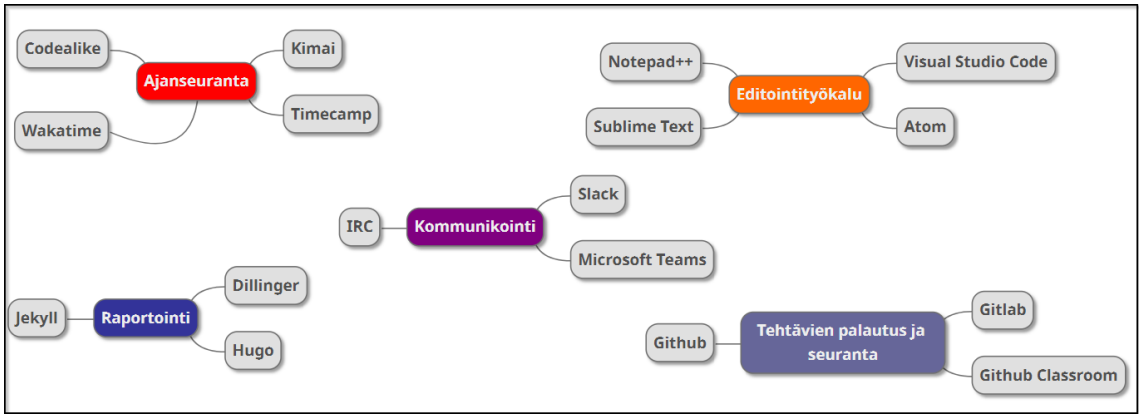
Automatisoitu editointityökalu arvostelee tehtävät automaattisesti, jonka jälkeen opiskelija voi jatkaa seuraaviin tehtäviin. Kaikkia kolmea mainittua palvelua yhdistää tehtävien palautusten automatisointi. Tehtävät palautetaan palvelun kautta, ja se suorittaa automaattisen testauksen, jolloin opiskelija saa suoraa palautetta tehtävän suorituksesta. Jos järjestelmä ei tunnista vastausta oikeaksi, voi opiskelija yrittää palautusta uudelleen. Usein ohjelma sisältää myös jonkinlaisen virheilmoituksen, ja siten ohjaa käyttäjää suorittamaan tehtävän hyväksytyksi. Helsingin MOOC-kurssi käyttää tehtävien palautukseen erillistä ohjelmointieditoria, kun freeCodeCamp ja Codeacademy hyväksyvät tehtävien palautuksen suoraan palvelun internet sivulle eli käyttöliittymälle. Näiden osalta käyttäjä voi suorittaa tehtäviä mistä tahansa laitteesta, jossa toimii internetselain. Tehtävien automaattinen testaus mahdollistaa usean käyttäjän tukeman palvelun mutta ei mahdollista palautetta itse tehtävän tekemisen prosessista. Palvelulle riittää, että tehtävä on kirjoitettu editoriin oikein, eikä se esimerkiksi ota kantaa, miten opiskelija on päätynyt lopputulokseen tai onko ohjelmointikoodi jäsennelty järkevästi. (Helsinki MOOC 2018.)

3.1 Työkalujen valinta

Ohjelmointiympäristön tekninen toteutus on rakennettu hyödyntäen eri työkaluja, jotka on suunniteltu toimivan toisiinsa nähden yhdessä luoden opiskelijalle ohjelmointityöskentelyä tukevan opiskeluympäristön. Rakennettavan ohjelmointiympäristön tekninen toteutus voidaan jakaa viiteen osaan:

- raportointi
- editointi
- ajan seuranta
- kommunikointi
- tehtävien palautus ja seuranta.

Nämä osa-alueet muodostavat kokonaisuuden, joka auttaa opiskelijaa ohjelmointiopiskelussa motivoiden ja jäsennellen opiskeltavan tiedon, sekä auttaa kurssin järjestäjää suoriutumaan hallinnollisista tehtävistä tehokkaammin. Raportointityökalu toimii kurssin materiaaliin tutustumisessa sekä tehtävien palautuksessa. Editointityökalulla voidaan kirjoittaa kurssin aiheena olevaa ohjelmointikoodia. Ajan seurannan avulla opiskelija kykenee havainnoimaan, mihin tehtävään tai aiheeseen kului eniten aikaa ja mitkä aihealueet olivat mahdollisesti työläitä. Reaaliaikainen kommunikointi auttaa opiskelijoita käymään yhdessä tai ohjaajan kanssa läpi ajatuksia tai ongelmia aiheuttavia pulmia. Tehtävien palautusten ja seurannan kautta opiskelijat voivat seurata etenemistään sekä kurssin järjestäjä voi hallinnoida opiskelijoita kurssiin liittyvissä asioissa. Näiden toimenpiteiden suorittamiseen on olemassa eri ohjelmistoja, joista osa on yhteensopivia toistensa kanssa, ja osassa on parempi tuki eri käyttöympäristöille.



Kuva 4. Mahdollisia sovelluksia käytettäväksi ohjelmointiympäristössä

Sovellusten valinnassa on huomioitu seikkoja, jotka vaikuttavat soveltuvuuteen tietojenkäsittelyn opiskelijoille. Kriteereitä valittaville sovelluksille ja työkaluille olivat muun muassa:

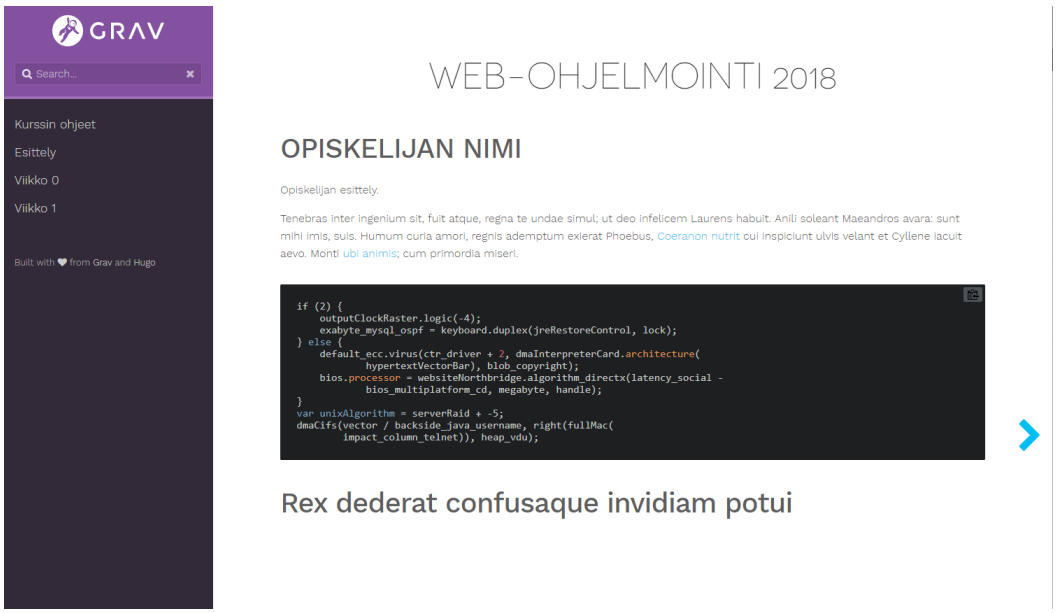
- vapaa käyttö tai ilmainen lisenssi
- tuki yleisimmille käyttöjärjestelmille ja laitteille
- mahdollisuus kustomointiin
- yhteensopivuus muiden ohjelmistojen kanssa
- mahdollinen käyttö työelämässä.

Erilaisia toteutuksia ja palveluita on markkinoilla useita, jotka sopisivat osittain tai kokonaan aiempien kriteerien perusteella työkaluksi ohjelmointiympäristöön. Lopullinen ratkaisu on tehty sovelluksen suosion perusteella, ja siihen on vaikuttanut myös se, kuinka hyvin valmistajat ovat tukeneet eri käyttöympäristöjen mahdollisuutta. Näitä työkaluja tai vastaavia ratkaisuja käytetään myös yleisesti ohjelmointiyrityksissä maailmanlaajuisesti. Teknologia ja työkalut kehittyvät jatkuvasti, joten on vaikeaa tai lähes mahdotonta ennustaa kuinka kauan valmistaja antaa tukea tuotteelleen tai kuinka teknologia-ala kehittyy. Koska ohjelmointikurssit uudistavat materiaaliaan suhteellisen usein, on mahdollista, että ohjelmointiympäristössä käytettävien työkalujen tarpeellisuutta tulee tarkastella uudistuksen yhteydessä. Yhden vuoden sisällä markkinoille on voinut tulla uusi parempi ratkaisu, jonka ottaminen osaksi ohjelmointiympäristön kokonaisuutta on tarpeellista, jos edellisen työkalun tuki on päättynyt. Ohjelmistoympäristöön valittujen työkalujen osalta valmistajan tarjoama tuki on hyvä tällä hetkellä, ja ne soveltuvat ominaisuuksiensa puolesta

ohjelmointiopiskelun tarpeisiin. Seuraavassa vaiheessa esitellään ohjelmointiympäristön käyttämät työkalut. Työkalujen näyttämä materiaali käyttää keksittyä ”Web-ohjelmointi 2018”-nimellä olevaa kurssia, joka ei ole oikea varsinainen ohjelmointikurssi. Materiaali on luotu demonstroimaan työkalujen mahdollisuutta esittää kurssimateriaalia opiskelijoille.

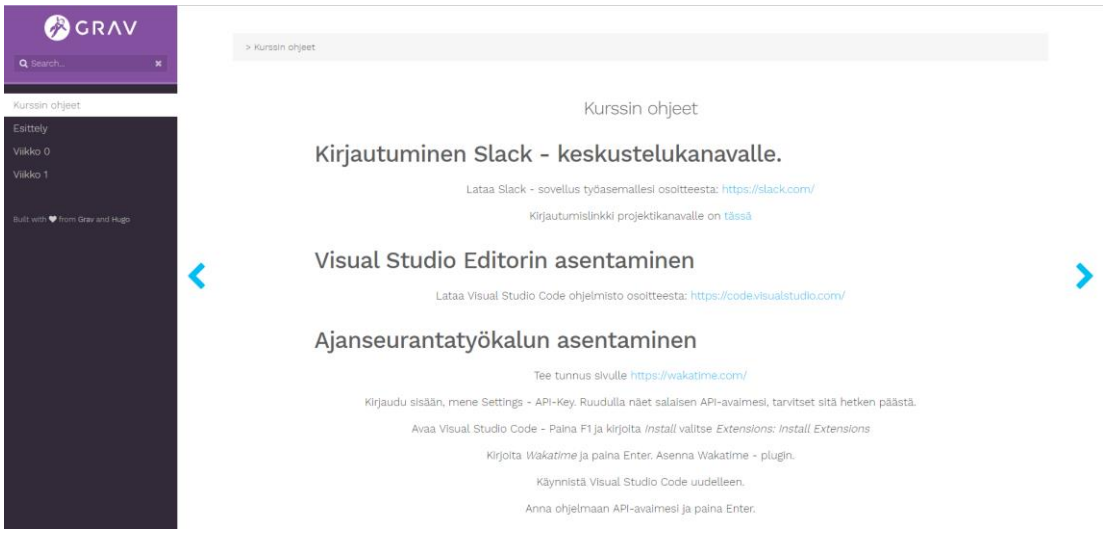
3.2 Raportointi

Raportointityökaluna ohjelmointiympäristössä käytetään ilmaista avoimen lähdekoodin ohjelmistokehystä nettisivujen sisällön luontiin nimeltä Hugo. Hugo mahdollistaa nopean ja mukautuvan ympäristön raportoinnin julkaisemiselle nettisivujen muodossa (Hugo 2018). Työkalu toimii web-sovelluksena, jota käytetään selaimessa. Sovelluksella muodostetaan eräänlainen kotisivu, jota kautta opiskelija voi tutustua kurssimateriaaliin sekä ylläpitää kurssipäiväkirjaa. Sisällön kirjoittamiseen käytetään Markdown-merkintäkieltä, joka on yleisesti käytössä useilla eri palveluilla. Sovelluksen käyttö ei vaadi käyttäjätunnuksen rekisteröimistä.



Kuva 5. Opiskelijan raportointisivun päänäkymä.

Hugo asennetaan käyttäjän tietokoneelle. Asennuksessa luodaan käyttäjän työasemalle paikallinen tiedosto ohjelmistosta, jota kautta voidaan käynnistää raportointityökalu. Sovellus avataan käyttäjän selaimessa, josta pääsee tutustumaan kurssimateriaaliin, kuten opetusmateriaaliin ja viikkotehtävään. Koska sivusto on lokaalisti käyttäjän tietokoneella, on opiskelijalla mahdollisuus muokata sivuston ulkoasua haluamallaan tavalla. Hugo tukee myös erilaisten valmiiden ulkoasujen käyttöä sovelluksessa. Tässä esimerkissä opiskelijat käyttävät valmiiksi määriteltyä ulkoasua nimeltä ”Learn”. Esimerkissä kurssin sisältö on jaoteltu vasemmalle eri välilehdille. Kurssi etenee viikoittain, joista jokainen viikko on oma kokonaisuutensa.



Kuva 6. Esimerkkisivu kurssin ohjeistuksesta

Esimerkkikuvassa ”Viikko 0” välilehdellä on esimerkkitehtäviä, joita opiskelija lähtee suorittamaan käydessään läpi kurssin materiaalia. Sovellus ei mahdollista vastausten antamista suoraan sovelluksessa, vaan ne tehdään erillisellä editorilla, joka myös asennetaan käyttäjän työasemalle.

Kurssimateriaali on jäsennelty niin, että opiskelija käy läpi vaiheittain teoriaa sekä käytäntöä. Esimerkkikuvassa 7a opiskelijalla on edessään kuvaus ohjelmointitehtävästä, joka suoritetaan palautettavana tehtävänä. Opiskelija lukee sivulta tehtävänannon sekä suorittaa tehtävän tekemällä sen ohjelmointieditoriin. Tehtävänannossa voi esimerkiksi olla malli ohjelmalle tai alkutilanne, jota opiskelija lähtee ratkaisemaan.

Kuva 7a. Esimerkkitehtävä, sekä vinkki, joka on piilotettu

Suoritetut tehtävät palautetaan arvioitavaksi myöhemmin joko liittämällä ne raportointiin tai erillisenä tiedostona. Jos opiskelija kohtaa tehtävässä ongelmia, kurssin ohjaaja on voinut asettaa tehtävien yhteyteen vinkkejä, joita voi halutessaan käyttää hyväksi tehtäviä suorittaessa.

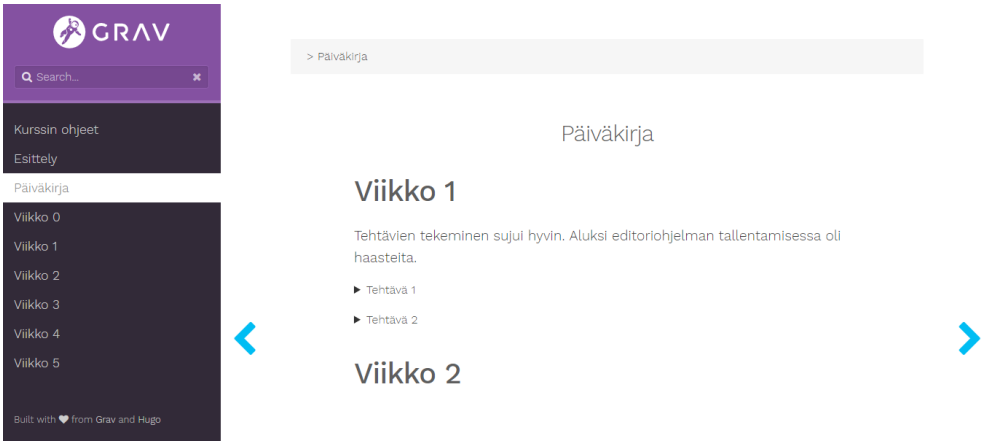
Kuva 7b. Esimerkkitehtävä, jonka vinkki on avattu

Vinkki avataan klikkaamalla tekstistä, jolloin avautuu näkymä esimerkkivastauksesta. Tällaisen ominaisuuden avulla opiskelijan huomiota voi kurssin suunnittelijan toimesta ohjata. Isoja kokonaisuuksia opetellessa, ei ole tarvetta esittää kaikkea saatavilla olevaa materiaalia heti kerralla.

Kurssimateriaalin jakamisen, määrän tai tyylin suhteen ei ole rajoituksia. Raportointityökaluun voi lisätä hyperlinkkejä, kuvia, videoita, ääntä sekä tekstiä. Koska työkalussa käytettävä Markdown-merkintäkieli tukee HTML-tagien käyttöä, pystyy sovellusta muokkaamaan monilla mahdollisilla eri tavoilla. Esimerkissä kurssimateriaali on jaoteltu eri osa-alueisiin. Materiaalin jakamisen nähdään auttavan oppimisessa (Blerkom 2008). Tällöin opiskelu tulisi jakaa useaan eri hetkeen, jolloin jokaisen viikon materiaali julkaistaan erikseen.

Blerkomin (2008) mukaan materiaali on hyvä jakaa pieniin osiin. Tällöin oppimista helpottaa se, että opiskelijan ei tarvitse muistaa suurta määrää materiaalia samalla kertaa, vaan hän voi käyttää oppimaansa esimerkiksi suoritettavan tehtävän tai kysymyksen avulla (Blerkom 2008). Raportointityökalun avulla materiaalin voi jakaa osiin, jolloin luettavan materiaalin jälkeen opiskelijalle avautuu suoritettava tehtävä.

Sovelluksen kautta voi myös raportoida kurssin etenemisestä. Opiskelija voi kirjoittaa raportin joko viikkokohtaisesti tai koota ne erilliseen osioon. Raportin kirjoittaminen tapahtuu kirjoittamalla teksti suoraan sovelluksen tiedostoon, joka esimerkissä löytyy valmiiksi tehtynä työasemalla omasta kansiosta. Koska raportointi tehdään käyttäen hyväksi Markdown-merkintäkieltä, on myös opiskelijalla mahdollisuus jäsennellä raporttia erinäisin keinoin.



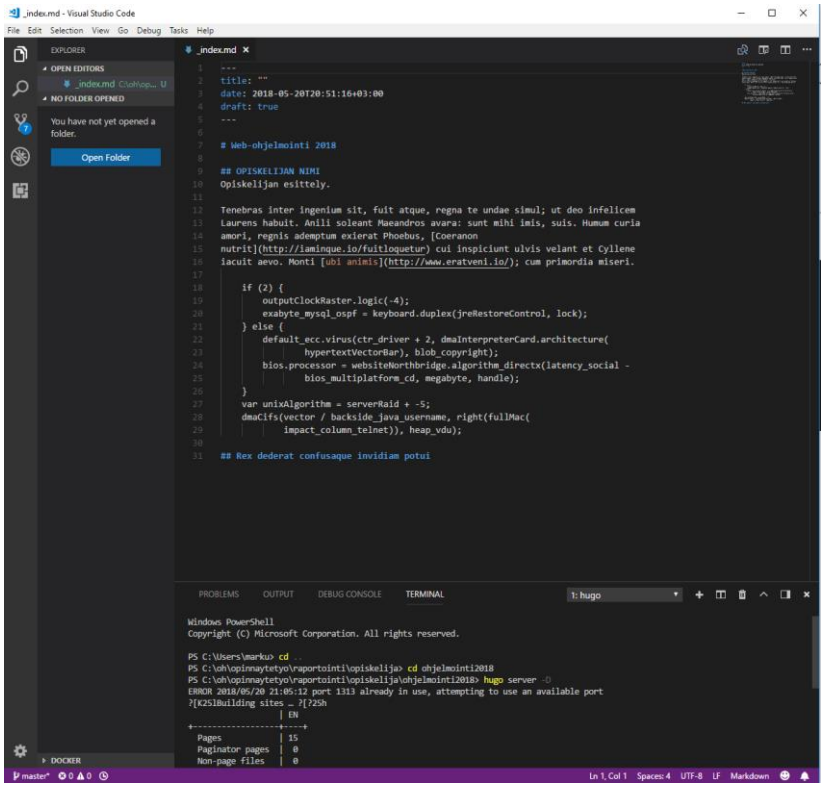
Kuva 8 Esimerkkikuva opiskelijan kirjoittamasta raportista sovelluksen käyttöliittymällä

Kuva 9 Esimerkkikuva opiskelijan kirjoittamasta raportista tiedostossa

Kuvassa 9 opiskelija on kirjoittanut tekstin valmiiksi annettuun tiedostoon Markdown- sekä HTML-merkintäkieltä käyttäen. Kurssin järjestävä ohjaaja on voinut luoda esimerkissä nähdyn valmiin raportointisivun, joka on jaettu opiskelijoille. Sivun luomisen voi myös antaa kurssin osallistujan vastuulle, jolloin hän rakentaa ja muodostaa sivuston itse. Tällöin kurssin materiaali ja tehtävä tulee jakaa kurssin osallistujalle muuta kautta. Tässä esimerkissä valmiiksi rakennettu raportointisivu sisältää kurssin materiaalin, joka on jaettu tehtävien palautukseen ja seurantaan käytetyn GitHub Classroom-palvelun kautta.

3.3 Editointi

Editointityökaluna käytetään Microsoftin julkaisemaa ilmaista, avoimen lähdekoodin Visual Studio Code-koodieditoria. Ohjelmointitehtävissä kyseisen editorin käyttö ei ole pakollista ohjelmoinnin toimivuuden kannalta, ja joissain erityisissä tilanteissa voi myös tarvita eri editorin käyttöä riippuen mihin ohjelmointikieleen kurssin materiaali perustuu. Visual Studio Code on valittu ohjelmointiympäristöön editointityökaluksi, koska se mahdollistaa lukuisten eri tiedostojen avaamisen ja muokkauksen ilmaiseksi. Ohjelman käyttö ei vaadi käyttäjätunnuksen rekisteröimistä.



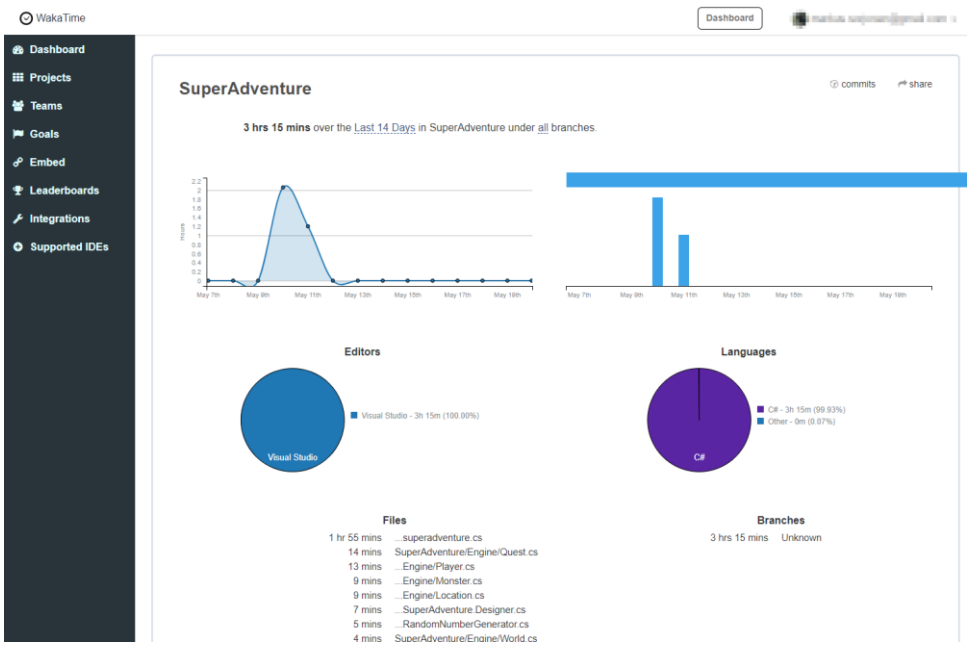
Kuva 10. Visual Studio Code, jossa raportointisovelluksen tiedosto avattuna

Editorissa on mahdollista avata raportointityökalun mukana tarvittavia tiedostoja sekä muokata niitä. Visual Studio Code mahdollistaa lukuisten eri tiedostojen muokkaamisen, ja on tarkoitettu yleiskäyttöiseksi ohjelmointieditoriksi. Ohjelman mahdollistama ilmaisten lisäosien käyttö myös kasvattaa mahdollisten ohjelmointikielien editoinnin suureksi. Kyseisestä editorista on myös mahdollista käyttää työaseman komentoriviä, jonka avulla ohjelmointikurssin tehtäviä voi halutessaan vastaanottaa sekä palauttaa.

3.4 Ajan seuranta

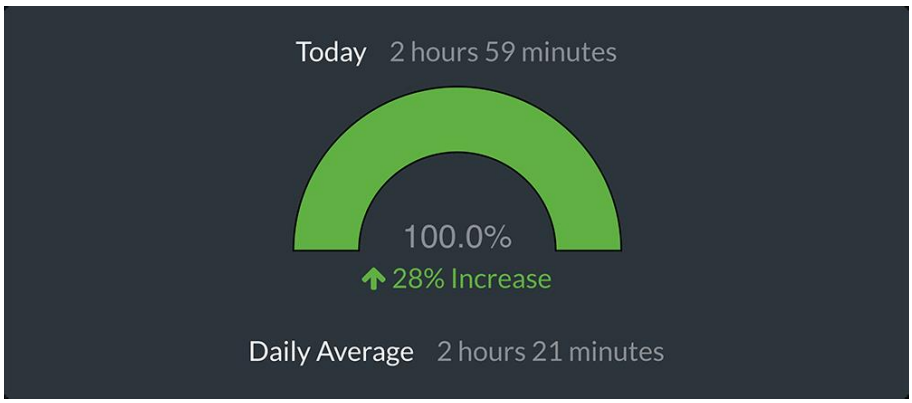
Ajanseurantaan käytetään muiden työkalujen kanssa yhteensopivaa Wakatime-palvelua. Wakatime-palvelu tarjoaa avoimen lähdekoodin liitännäisiä, joiden avulla voidaan seurata ohjelmointityöskentelyyn käytettyä aikaa. Liitännäinen asennetaan editoriin, jolloin se seuraa tiettyyn ohjelmointiprojektiin käytettyä aikaa. Palvelu tunnistaa automaattisesti projektin nimen, tiedoston, jota

editoidaan sekä käytettävän ohjelmointikielen. Palvelua käyttääkseen käyttäjän tulee rekisteröidä ilmainen käyttäjätunnus.



Kuva 11. Wakatimen etusivu, kun käyttäjä on kirjautunut sisään

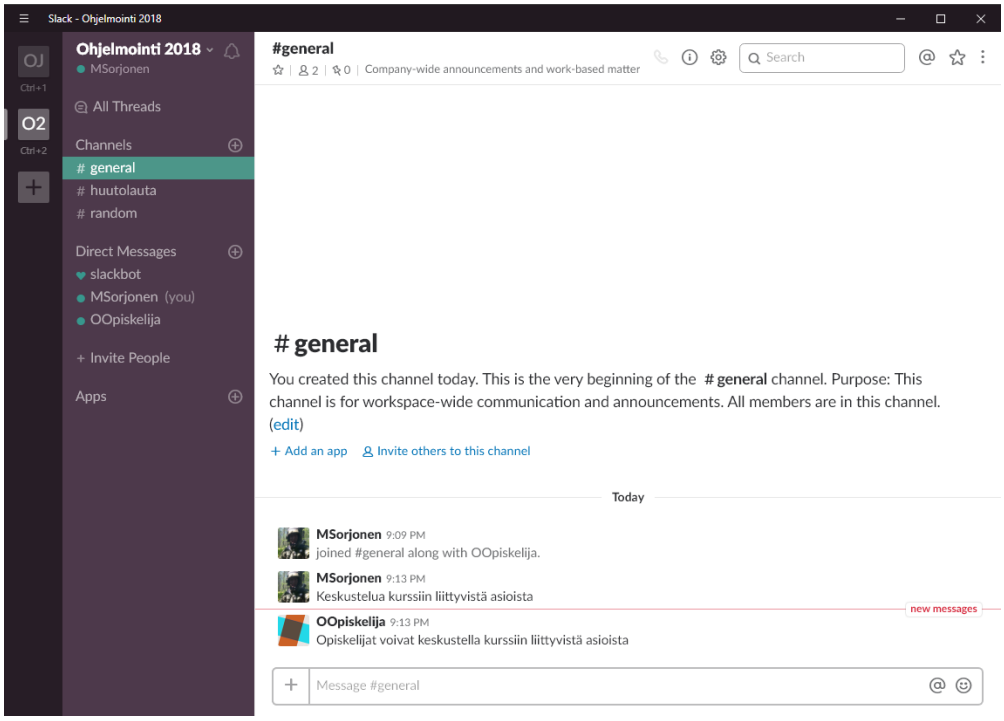
Palvelun web-sivulla opiskelija voi seurata eri tehtäviin tai projekteihin käyttämäänsä aikaa. Wakatime-palvelussa on myös mahdollista perustaa maksullinen tiimitila, jossa ryhmän jäsenet näkevät keskenään kuinka paljon eri projekteihin on käytetty aikaa. Palvelu tunnistaa ainoastaan ohjelmointieditorissa ohjelmointiin käytettyä aikaa. Ohjelma osaa jäsennellä eri editoreissa, eri projekteissa ja jopa eri ohjelmointikielillä käytetyn ajan. Palvelun ilmainen versio ei anna kootusti kattavaa dataa yli viikon ajalta.



Kuva 12. Kuva Wakatimen laskemasta ajasta (Kuva: Wakatime 2020.)

3.5 Kommunikointi

Kurssin osallistujien väliseen kommunikointiin käytetään Slack-nimistä ohjelmistoa, joka mahdollistaa ryhmien välisen keskustelun, yksityisviestien lähettämisen sekä integroinnin muihin työkaluihin. Ohjelma ladataan käyttäjän työasemalle. Palveluun voi luoda yksityisen työtilan, johon voi kutsua uusia käyttäjiä osallistumaan keskusteluihin. Ohjelmointikurssin opiskelijat voivat ohjaajan luomassa työtilassa keskustella kurssiin liittyvissä asioissa keskenään tai tarvittaessa ohjaajan kanssa. Palvelun kautta pystyy myös lähettämään tiedostoja, sekä sitä voi käyttää myös mobiililaitteilla.



Kuva 13. Ohjelmointikurssia varten luotu työtila

Ohjelman käyttö on pienille tiimeille ilmaista, ja maksullista suurille tiimeille sekä kaupalliseen toimintaan. Palvelu tukee myös eri ohjelmistojen, kuten versionhallintaohjelmistojen kanssa yhteensopivuutta. Tällaisessa tilanteessa palveluun voi saada automaattiviestin, jos osallistuja lataa uusia tiedostoja versionhallintaan, sinne lähetetään kommentti, tai ajastetun tehtävän määräaika

on päättymässä. Keskusteluryhmien luominen onnistuu pääkäyttäjän toimesta, sekä käyttäjät voivat lähettää toisilleen yksityisviestejä. Sovellus on saatavina myös puhelimille.

3.6 Tehtävien palautus ja seuranta

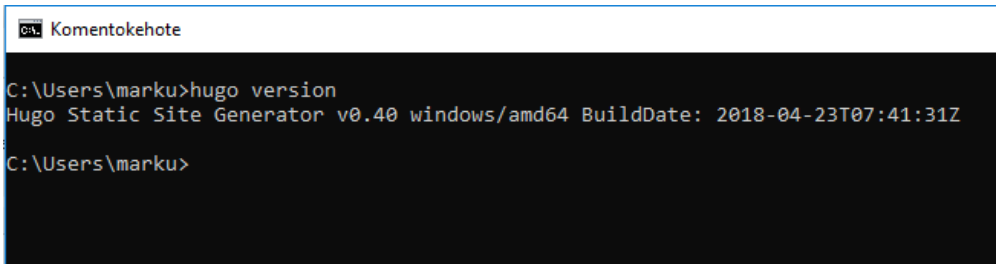
Kurssin ohjelmointitehtävien palautukseen käytetään GitHub-versionhallinta työkalun tarjoamaa GitHub Classroom-palvelua. Palvelussa on mahdollista luoda kurssin ohjaajan toimesta työtila kurssia ja sen tehtäviä varten. Palvelun kautta kurssin ohjaaja pystyy hallitsemaan kurssiin osallistuvia opiskelijoita, antamaan sekä vastaanottamaan opiskelijoiden palautettavia ohjelmointitehtäviä.
Palvelu on ilmainen ja tarkoitettu ensisijaisesti koulujen käyttöön. Palvelun käyttö vaatii ilmaisen käyttäjätunnuksen rekisteröinnin. Palvelussa muodostetaan organisaatiolla virtuaalinen luokkatila, jonka kautta kurssin järjestäjä voi jakaa osallistumislinkin opiskelijoille. Tämän avulla opiskelija tallentaa oman säilytystilan GitHub-palveluun, josta tulee säilytystila kurssin tai yksittäisen tehtävän palautuksille. Palvelu mahdollistaa myös yksilö- sekä kurssitehtävien muodostamisen. Luokkatila on pääkäyttäjän hallittavissa, ja tehtävien palautus on myös mahdollista asettaa salaiseksi, jolloin ainoastaan sallitut käyttäjät pystyvät näkemään palautetut tehtävät.

4 Ohjelmointiympäristön työkalujen asentaminen ja käyttö

4.1 Raportointityökalu

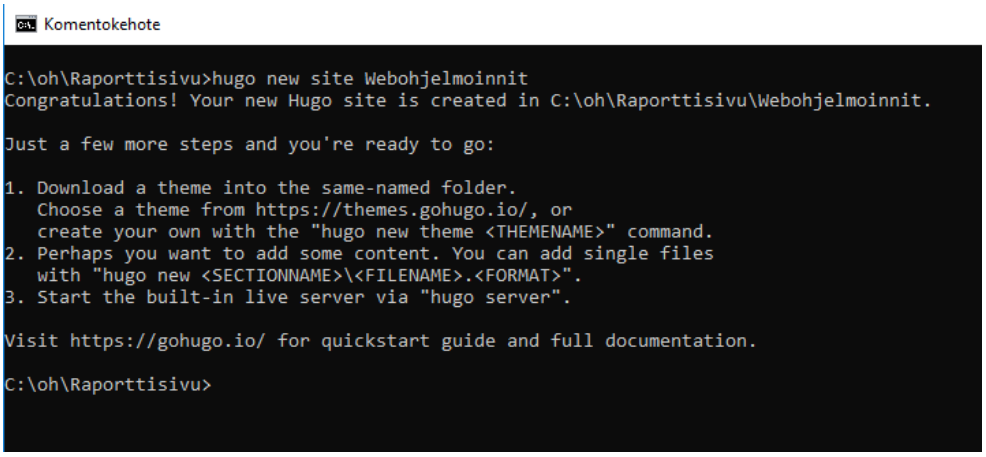
Ohjelmointikurssin raportointia varten muodostetaan valmiiksi pohja käyttäen hyväksi Hugo-palvelua. Palvelu muodostaa valmiin selaimessa toimivan sivuston, johon sisällön ja tyylin voi määritellä itse. Hugo täytyy ladata käyttäjän työasemalle. Latausohjeet löytyvät eri käyttöjärjestelmille valmistajan kotisivuilta, osoitteesta https://gohugo.io/getting-started/installing.

Ladattuja tiedostoja käytetään tietokoneen komentoriviltä. Tarkastaaksesi onko Hugo asennettu työasemalle, voit ajaa komentorivissä käskyn ”hugo version” (kuva 14). Jos käsky palauttaa Hugon versionumeron, se on asennettu työasemalle.



Kuva 14. Windows käyttöjärjestelmän komentokehote, johon annettu käsky ”hugo version”

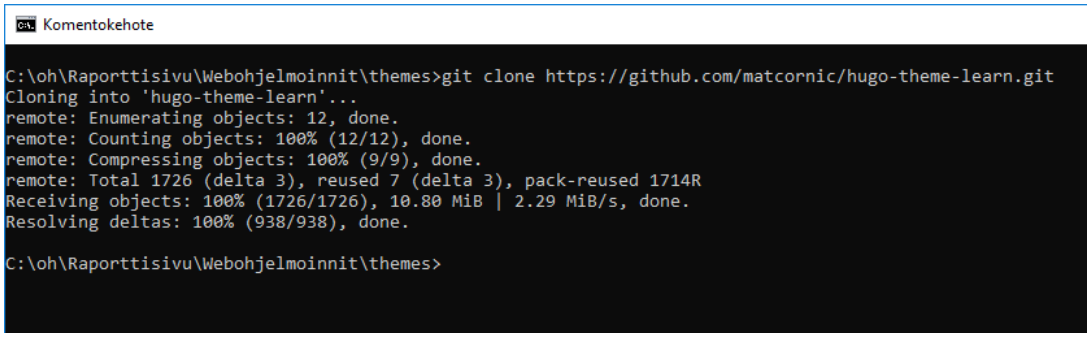
Sivun luomista varten komentorivillä ajetaan komento ”hugo new site Web-ohjelmoinnit”, jossa ”Web-ohjelmoinnit” on sivuston nimi (kuva 15). Komentokehote ilmoittaa, jos sivuston tiedostot on luotu onnistuneesti.



Kuva 15. Komentokehotteelta luodaan uusi sivusto

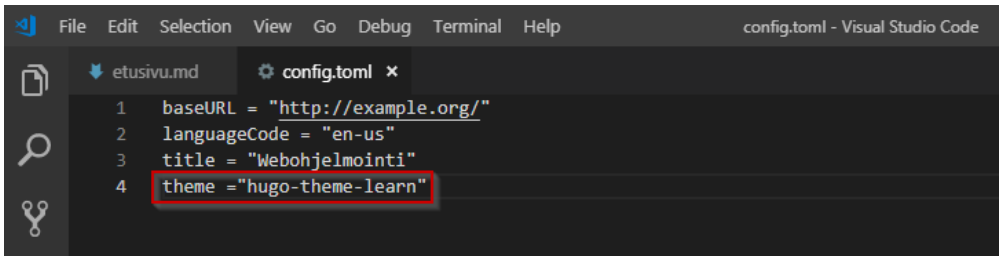
Komentokehote ohjeistaa myös, kuinka sivuston rakentamisessa voi jatkaa eteenpäin. Tässä vaiheessa työaseman latauskohteesta löytyy tiedostot Hugo-sivuston luomiseen. Sivustoon voi halutessaan asettaa valmiin teeman. Teeman voi hakea internetsivulta https://themes.gohugo.io/. Tässä esimerkissä käytetään teemaa nimeltä ”Learn”.

Teema asennetaan navigoimalla komentokehotteessa juuri luodun sivuston kansioon ”Themes”, johon haluttu teema asennetaan komennolla ”git clone https://github.com/matcornic/hugo-theme-learn.git” (kuva 16).



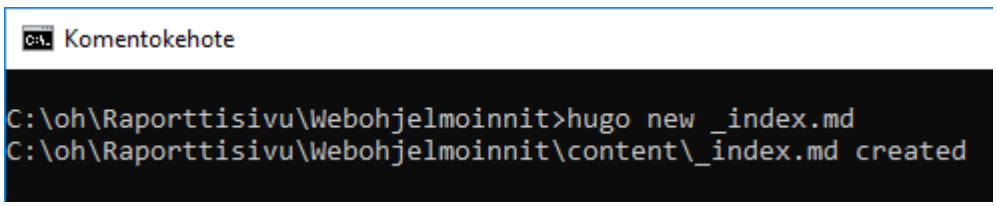
Kuva 16. Komentokehotteelta asennetaan teema sivustolle

Asennettuun teemaan tulee viitata ”config.toml”-nimisessä tiedostossa, joka löytyy sivuston kansiosta. Jos teeman asennus ei ole tehnyt sitä automaattisesti, se tulee tehdä manuaalisesti. Kuvassa 16 on lisätty kyseiseen tiedostoon teksti ”theme = ’hugo-theme-learn’ ” joka on esimerkissä asennetun teeman nimi.



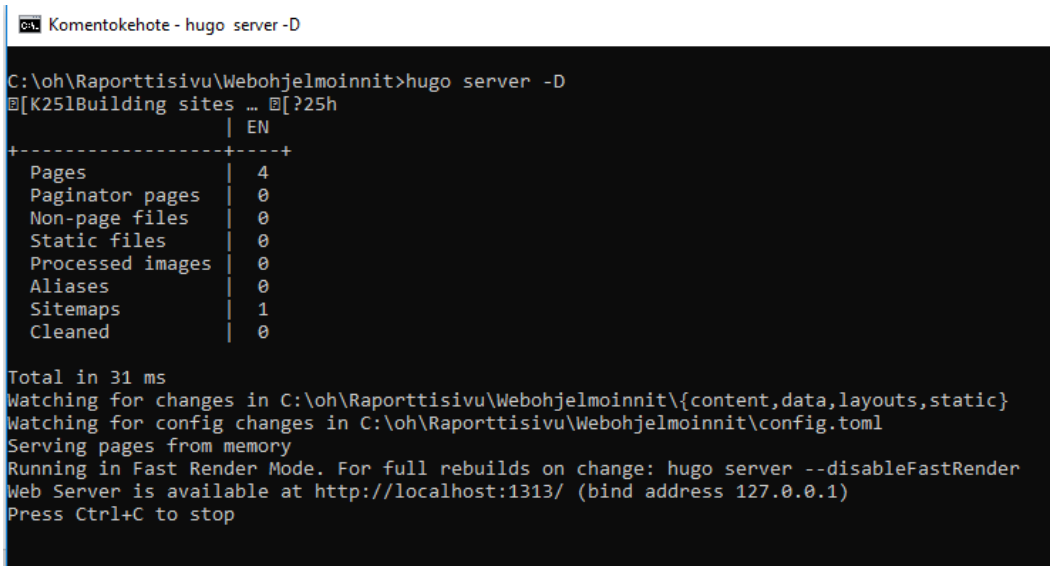
Kuva 17. Lisätty viittaus asennettuun teemaan

Luodaan ennen sivuston tarkastelua etusivu. Uuden sivun luominen onnistuu komentokehotteella komennolla ”hugo new _index.md”, jossa ”_index.md” on luodun sivun tiedostonimi (kuva 18).



Kuva 18. Komentokehotteella luodaan uusi sivu

Sivun luonnin jälkeen sitä voi editoida haluamallaan tavalla. Tässä vaiheessa käyttäjä voi luoda sisällön sivustolle, joka näytetään raportointityökalussa. Sivua editoidaan menemällä työasemalla kansioon, johon se on asennettu, ja avaamalla Markdown – tiedosto ediointia tukevalla ohjelmalla. Esimerkiksi Visual Studio Code – editori tukee Markdown-tiedostojen muokkausta. Ennen käyttöä sivusto tulee myös käynnistää. Se onnistuu komentokehotteella komennolla ”hugo server -D” (kuva 19). Oletuksena sivusto toimii osoitteessa http://localhost:1313/. Sivustoon voi mennä menemällä edellä olevaan osoitteeseen internetselaimella.

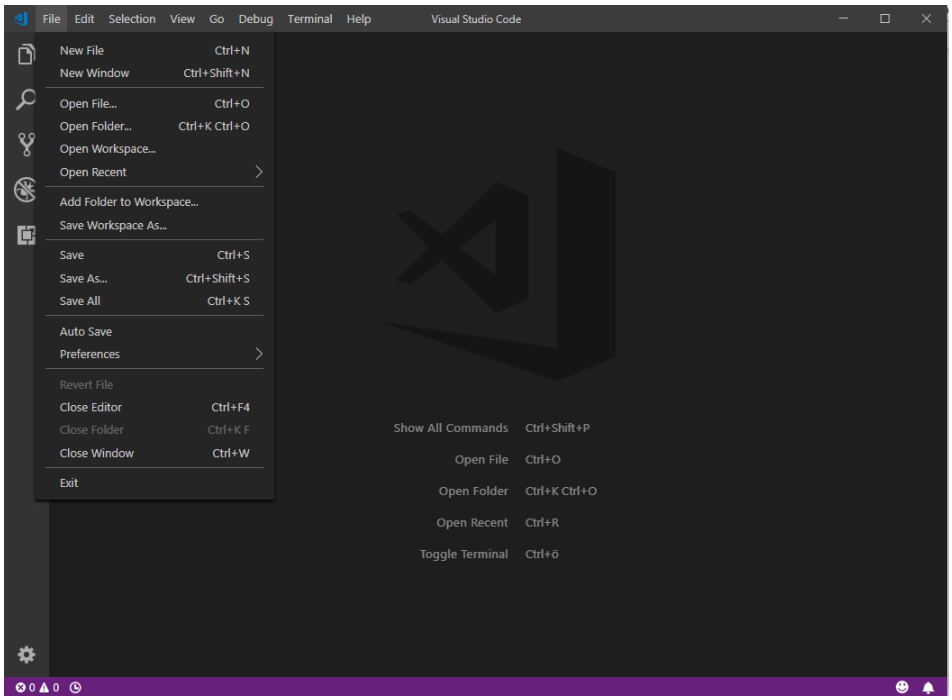


Kuva 19. Sivusto käynnissä. Kuvassa esimerkki, johon on luotu useampi sivu

4.2 Editointityökalu

Tässä projektissa käytetään avoimen lähdekoodin editoria Visual Studio Code, joka on Microsoftin julkaisema sekä toimii useimmilla käyttöjärjestelmillä. Editorilla on mahdollista avata ja muokata paljon eri ohjelmointikielen tiedostoja. Ohjelma ladataan käyttäjän työasemalle osoitteesta: https://code.visualstudio.com.

Editorissa on mahdollista virheenjäljitys sekä Git-komentojen käyttö. Ohjelmaan on saatavilla lukuisia työskentelyä helpottavia liitännäisiä, joita voi ladata tarpeen mukaan sovelluksen kautta ilmaiseksi. Ohjelman asentamisen jälkeen Visual Studio Codessa voi avata tiedostoja valitsemalla vasemmasta ylälaidasta ”File” ja ”Open file”, jonka jälkeen tiedoston voi etsiä käyttäjän työasemalta (kuva 20).

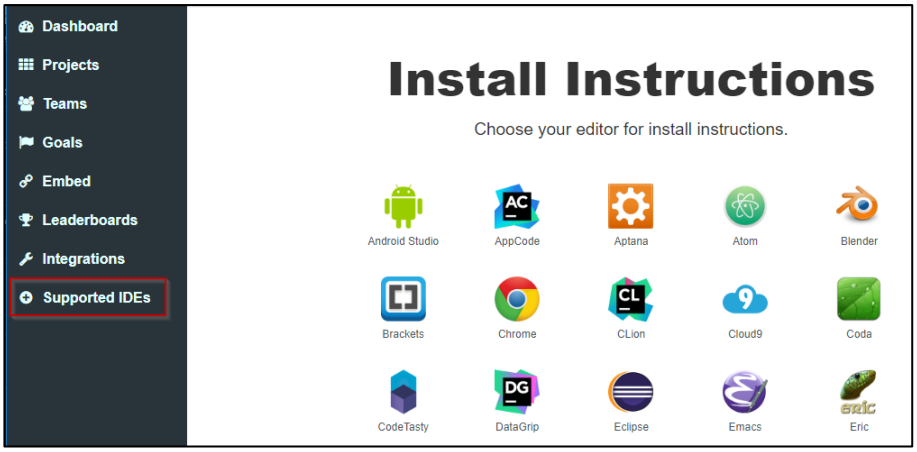


Kuva 20. Tiedoston avaaminen Visual Studio Codessa

4.3 Ajan seurantatyökalu

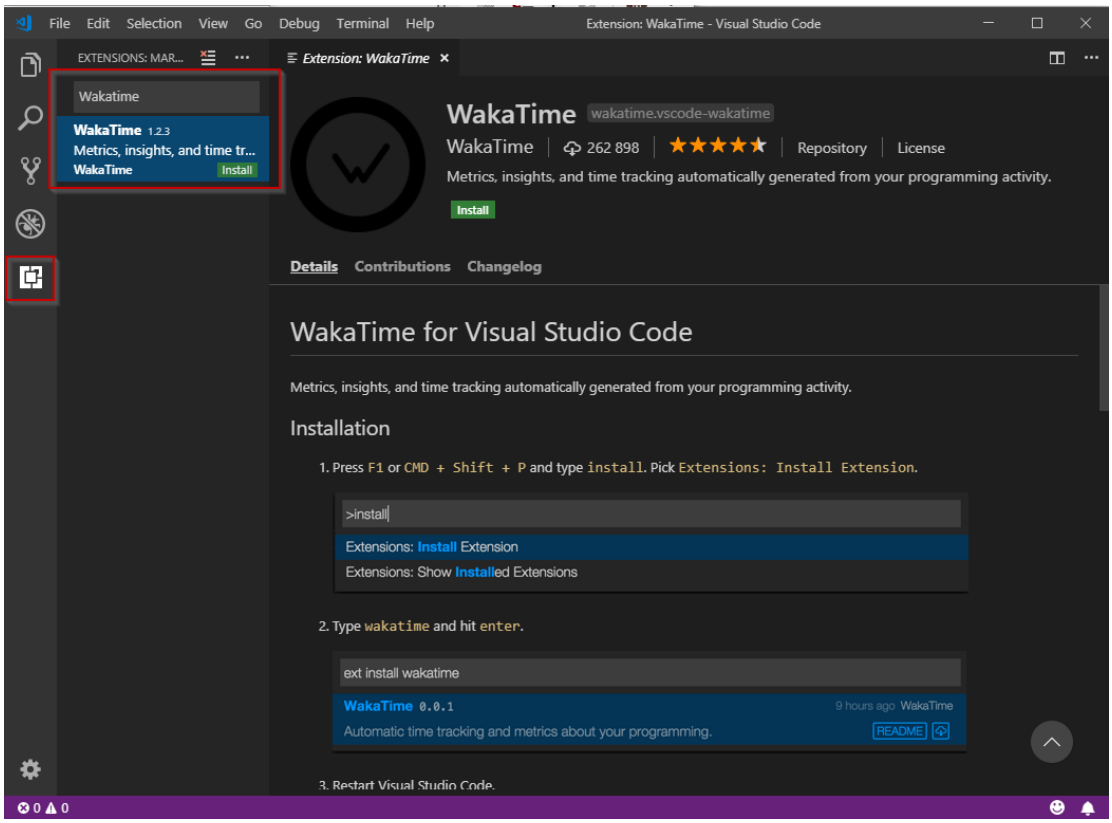
Ajanhallintaan käytetään ilmaista Wakatime-palvelua, jonka käyttöä varten tulee rekisteröidä ilmainen käyttäjätunnus. Ilmaisen käyttäjätunnuksen voi rekisteröidä osoitteessa: https://wakatime.com/signup.

Rekisteröinnin jälkeen käyttäjä voi tarkastella eri ohjelmointiprojekteihin käytettyä aikaa. Sovellus laskee ohjelmointiin käytetyn ajan. Ajan laskemista varten käyttäjän ohjelmointieditoriin tulee asentaa Wakatimen liitännäinen, jotta tieto tallentuu palveluun. Wakatime tukee useimpia ohjelmointieditoreita (kuva 21).



Kuva 21. Wakatimen tuetut integraatiot

Visual Studio Codeen liitännäisen asentaminen on tehty helpoksi. Kun ohjelma on avattuna, uusia liitännäisiä voi etsiä painamalla vasemmassa reunassa olevaa kuvaketta (kuva 22), tai valitsemalla ylälaidasta ”View” ja ”Extensions”. Hakemalla liitännäisten hakukentässä sanalla ”Wakatime” voi tarvittavan liitännäisen asentaa ohjelmaan painamalla ”Install”.



Kuva 22. Liitännäisen asennus

Asennuksen jälkeen ohjelma kysyy API-avainta, joka tulee antaa. API-avain on henkilökohtainen tunniste, jonka kautta Wakatime tunnistaa oikean käyttäjän. Avain löytyy kirjautumalla Wakatime-palveluun, menemällä ”Settings – Account” välilehdelle ja katsomalla avain kohdasta ”Secret API-key” (kuva 23).

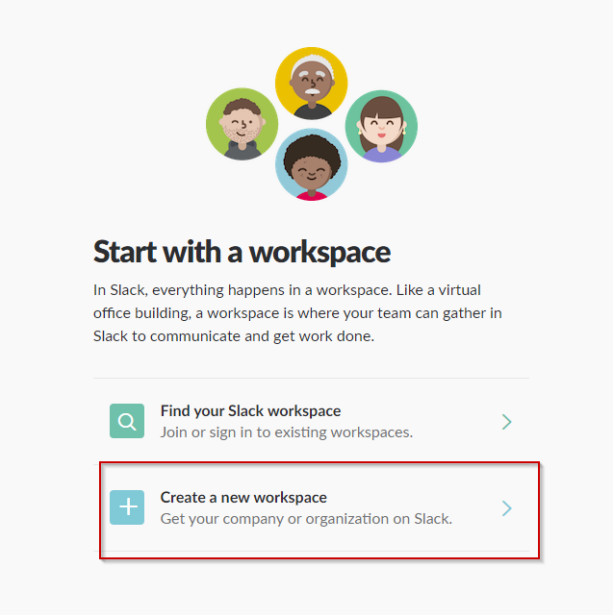
Kuva 23. Wakatimen API-avain

4.4 Kommunikointityökalu

Kommunikointiin käytetään Slack-palvelua. Palvelu mahdollistaa ilmaisen keskustelutilan pienille projekteille ilmaiseksi. Palvelua voi käyttää ilman erikseen ladattavaa ohjelmistoa selaimessa, mutta suositeltavaa on asentaa ohjelma työasemalle. Sen voi ladata ilmaiseksi osoitteesta: https://slack.com/downloads/

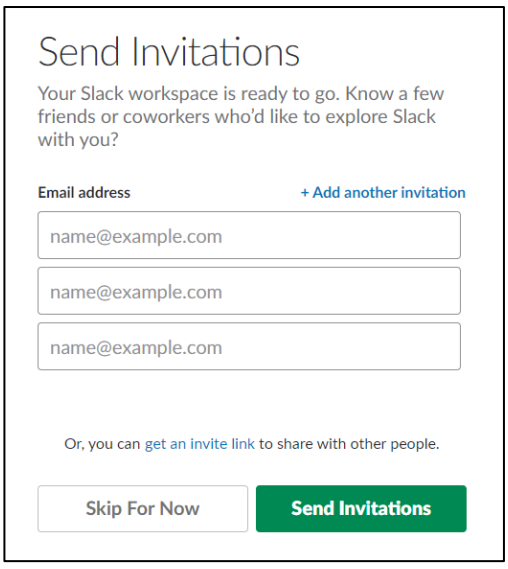
Slack-keskustelutiloihin kirjaudutaan erikseen käyttäjätunnuksen perusteella. Uusi keskustelutila ja käyttäjätunnus muodostetaan menemällä sivuilla kohtaan ”Get-started”.

Valitsemalla ”Create a new workspace” luodaan uusi työtila projektille (kuva 24).



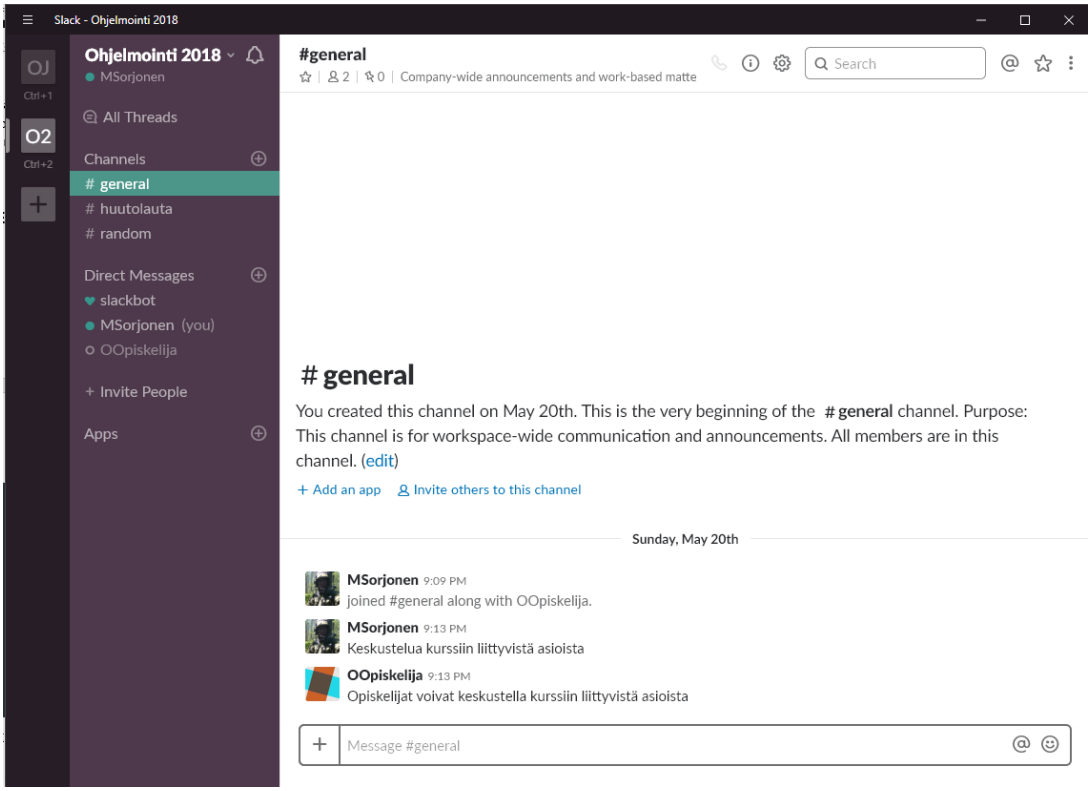
Kuva 24. Luo uusi työtila

Tämän jälkeen palveluun annetaan haluttu sähköpostiosoite, jonka kautta palvelu pyytää asettamaan viestissä lukevat 6 tunnistetta näytölle. Myöhemmin palvelulle annetaan oma nimi, jonka perusteella muut käyttäjä tunnistavat sinut. Käyttäjätunnuksen luonnin jälkeen valitaan projektille nimi sekä osoite, jota kautta muut pääsevät kirjautumaan projektille luotuun keskustelutilaan. Luonnin jälkeen on mahdollista lähettää kutsuja sähköpostin välityksellä muille osallistujille (kuva 25).



Kuva 25. Kutsujen lähettäminen

Kutsun saanut henkilö voi rekisteröidä oman tunnuksensa juuri luotuun työtilaan. Työtilassa vasemmalla voi lisätä vaihtoehtoisia kanavia keskusteluille. Automaattisesti Slack muodostaa ”#general” sekä ”#random” nimiset keskustelukanavat, johon käyttäjät voivat vapaasti lähettää viestejä. ”Direct messages” kohdassa näkyy ketkä käyttävät sillä hetkellä palvelua, sekä heille voi lähettää yksityisviestejä (kuva 26).



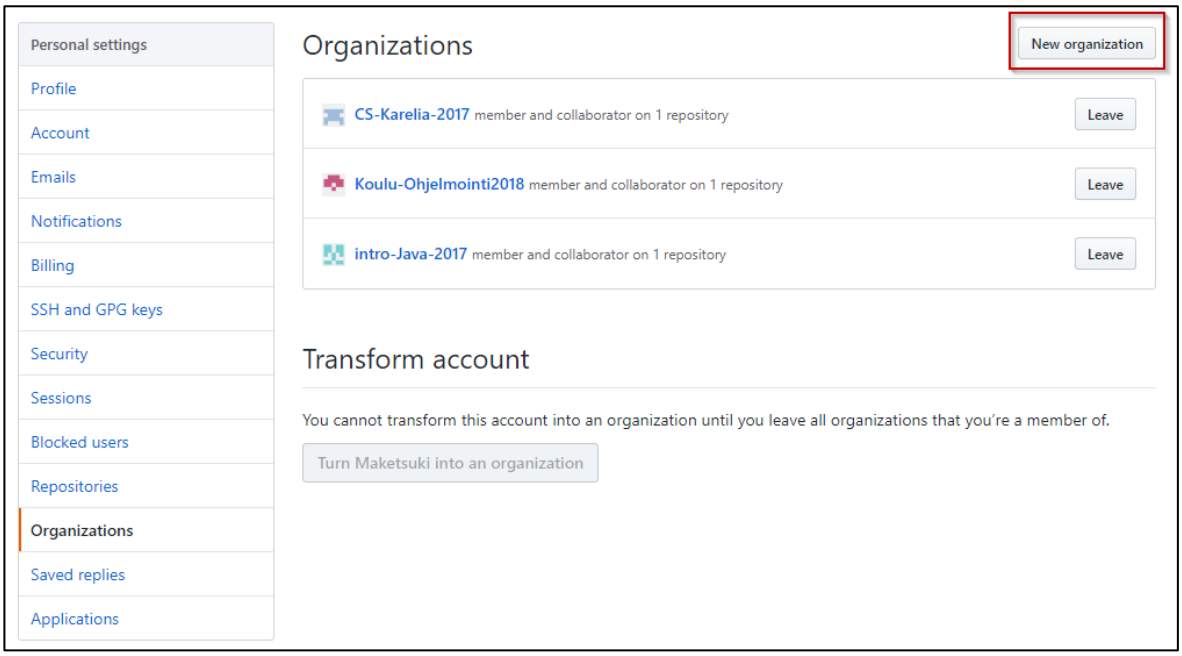
Kuva 26. Luotu työtila keskusteluille sekä keskusteluikkuna

4.5 Tehtävien palautuksen ja seurannan työkalu

Tehtävien jakamiseen sekä palauttamiseen kätetään GitHubin julkaisemaa GitHub Classroom-palvelua, joka on suunniteltu pienille sekä suurille ryhmille. Palvelun käyttö vaatii GitHub käyttäjätunnuksen, jonka voi rekisteröidä osoitteessa https://github.com/

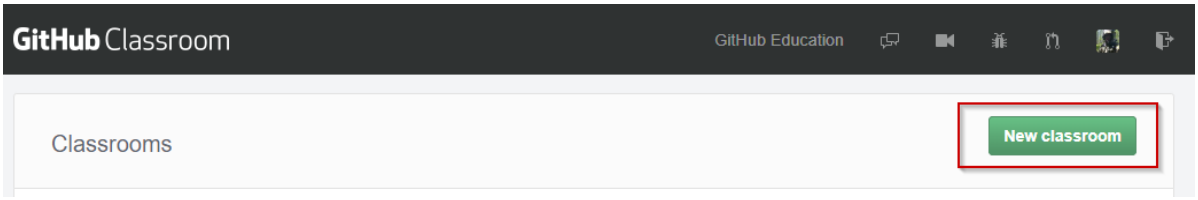
GitHub Classroom-palvelulla perustetaan virtuaalinen luokkatila, jota kurssin ohjaaja hallinnoi, ja jonka kautta opiskelijat voivat rekisteröidä oman version

työtilasta tehtävien palautusta varten. Ennen luokkatilan perustamista GitHub-palveluun tulee luoda organisaatio, johon luokat sisältyvät. Organisaatio luodaan kirjautumalla palveluun, menemällä ”Personal settings – Organizations” ja valitsemalla ”New Organization” (kuva 27). Organisaation luonnin yhteydessä valitaan nimi, asetetaan tiedot sekä halutessa kutsutaan muita käyttäjiä.



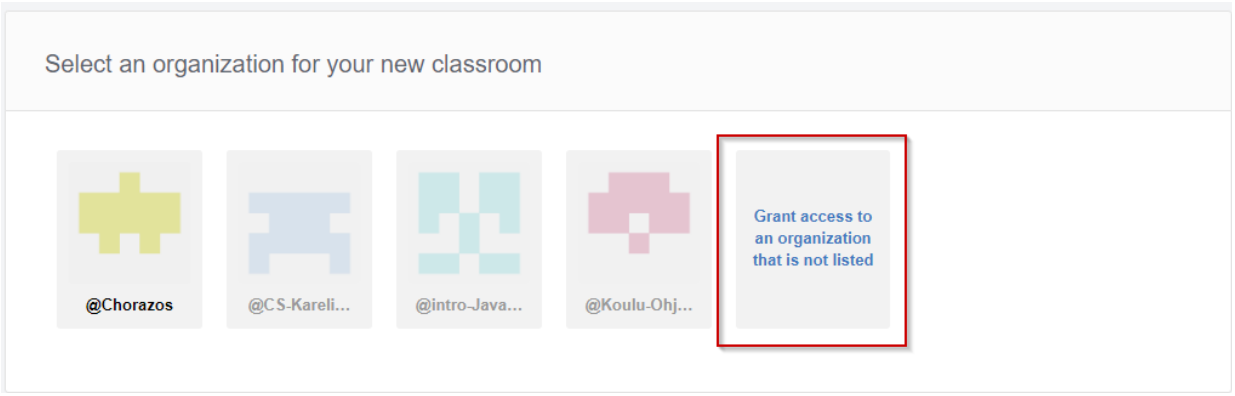
Kuva 27. Organisaation perustaminen

Luokkatilan perustaminen alkaa menemällä osoitteeseen: https://classroom.github.com/ Uusi luokkatila perustetaan valitsemalla GitHub Classroomin etusivulla oikeasta laidasta painikkeesta ”New Classroom” (kuva 28).



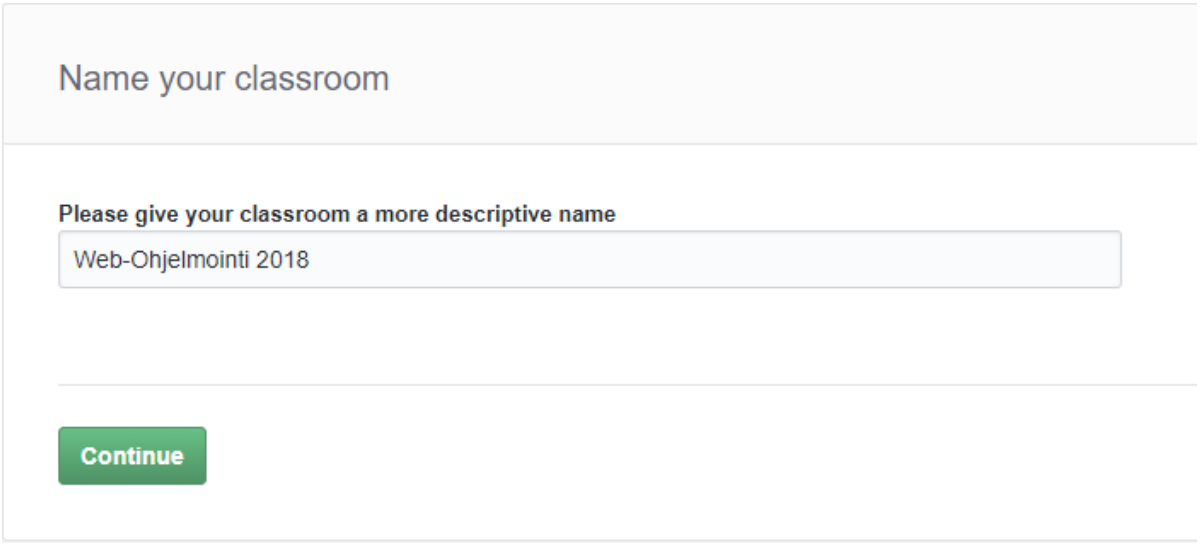
Kuva 28. Luokkatilan perustaminen

Seuraavaksi valitaan organisaatio, jonka yhteyteen uusi luokkatila perustetaan. Jos oikea organisaatio ei ole valittavissa, valinnalla ”Grant access to an organization that is not listed” voidaan lisätä se listalle (kuva 29).



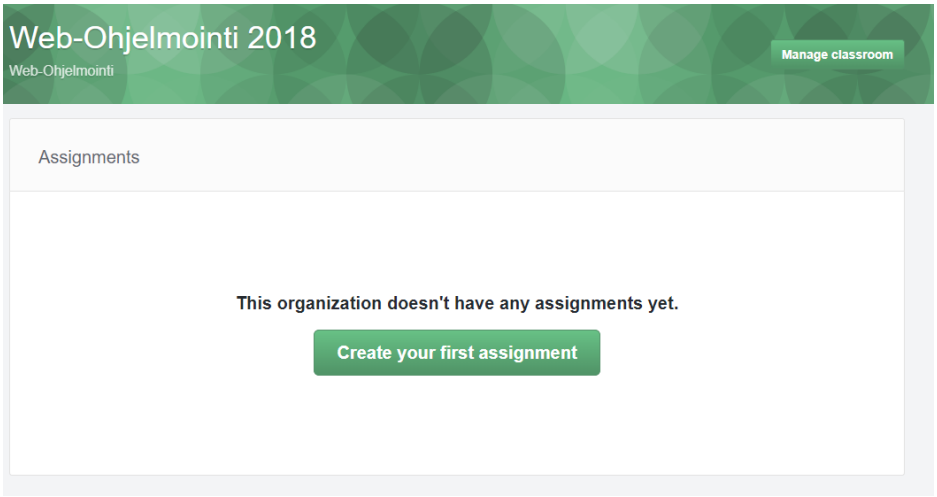
Kuva 29. Organisaation valinta

Organisaation valitsemisen jälkeen annetaan virtuaaliluokalle nimi (kuva 30).



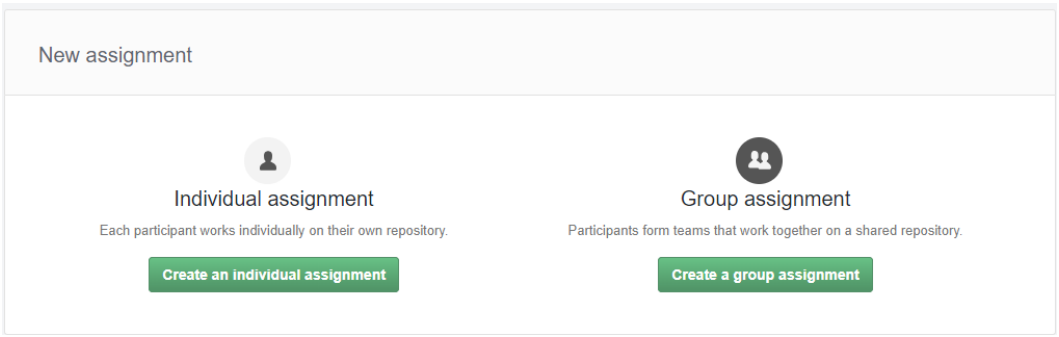
Kuva 30. Nimen antaminen

Sen jälkeen luokkatilalle voi antaa muita pääkäyttäjiä. Tällaiset käyttäjät voivat olla esimerkiksi kurssin muita järjestäjiä ja ohjaajia. Pääkäyttäjän tulee olla Owner-jäsen myös organisaatiossa, johon luokkatila on luotu. Luokkatilan muodostuksen jälkeen voi tehdä ensimmäisen kurssitehtävän (kuva 31). Se tehdään valitsemalla ”Create your first assignment”.



Kuva 31. Ensimmäisen kurssitehtävän luonti

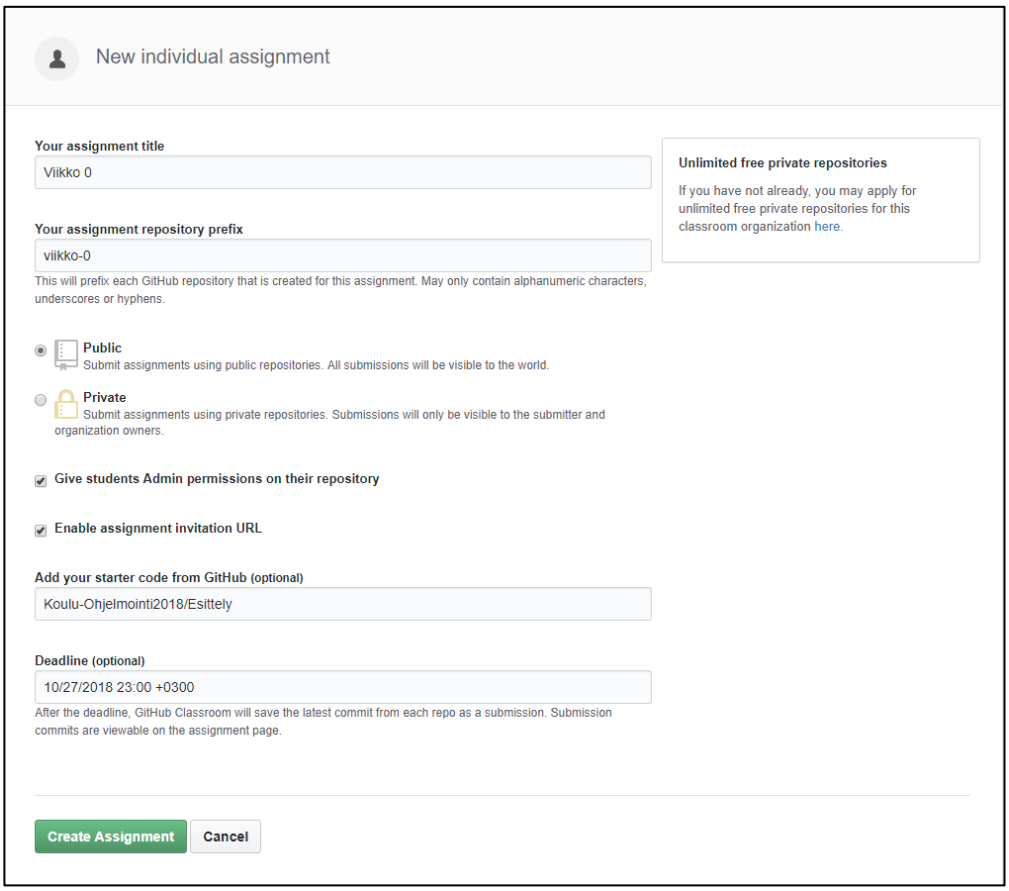
Tehtävän voi luoda joko yksilö – tai ryhmätehtäväksi. Yksilötehtävässä jokainen opiskelija tekee ja palauttaa tehtävän omaan säilytyspaikkaan. Tässä esimerkissä luodaan yksilötehtävä, valitsemalla ”Create an individual assignment” (kuva 32).



Kuva 32. Tehtävän luonti

Seuraavassa vaiheessa asetetaan tehtävän tiedot. Tehtävälle voi antaa nimen, onko tehtävä julkinen vai yksityinen, onko tehtävän pohjana jokin valmiiksi luotu säilytyspaikka ja onko tehtävällä viimeinen palautuspäivämäärä (kuva 33). Seuraavassa esimerkissä tehtävälle on annettu nimeksi ”Viikko 0”.

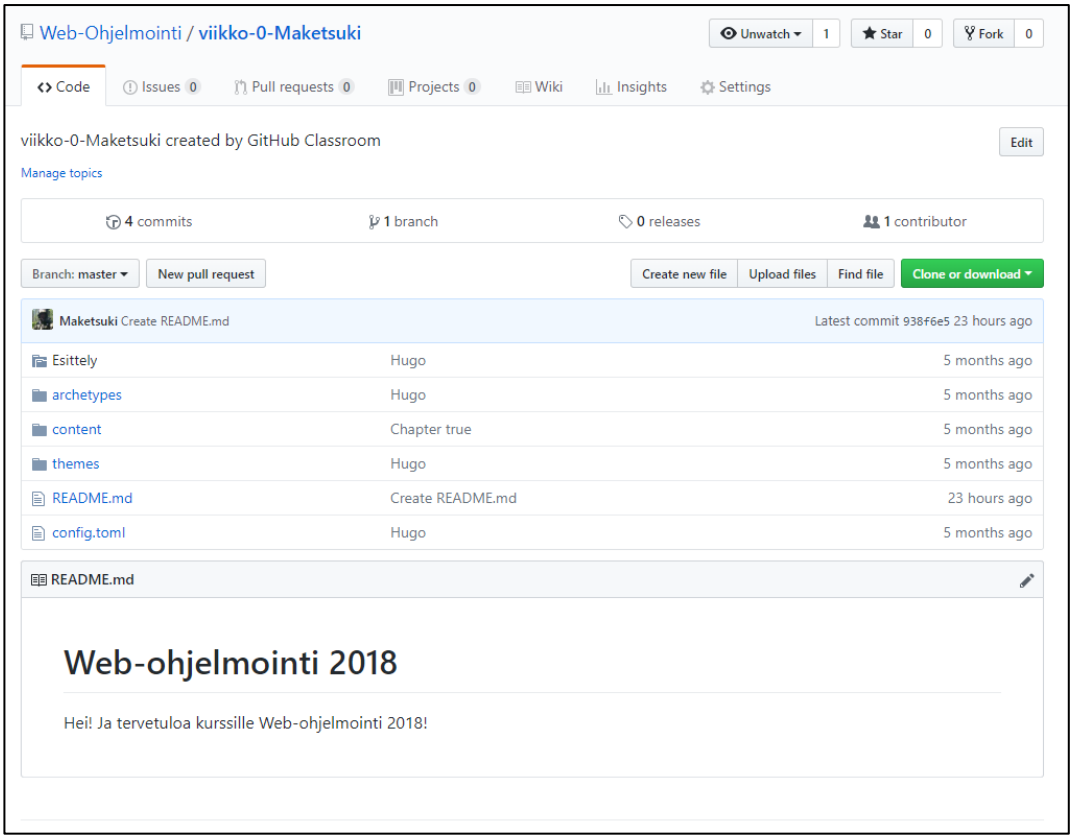
Esimerkissä käytetään aloituskoodina erikseen säilytyspaikkaan luotua Hugo-sivustoa. Aloituskoodi voi olla mikä tahansa ennalta asetettu säilytyspaikka. Pohjana voi esimerkiksi olla kurssin ohjeet, raportointityökalu, ohjelmointitehtävät tai kaikki edellä mainitut. Tehtävä luodaan valitsemalla ”Create Assignment”.



Kuva 33. Tehtävän tietojen luonti

Tehtävän luomisen jälkeen tehtävän voi jakaa opiskelijoille. Palvelu muodostaa tehtävästä linkin, jonka saatuaan opiskelijalle kopioituu kopio tehtävästä omaan GitHub-käyttäjätiliin. Opiskelijan näkökulmassa linkin käytön jälkeen palvelu kysyy, haluaako käyttäjä päästä luotuun tehtävään.

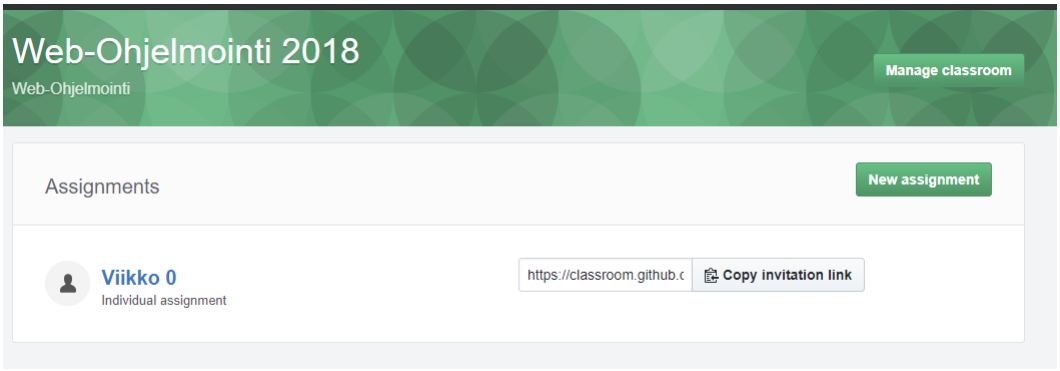
Hyväksymisen jälkeen palvelu muodostaa tehtävästä säilytystilan opiskelijalle, sekä lataa siihen ennalta määritellyn aloituskoodin, jos sellainen oli tehtävää luodessa asetettu. Opiskelija pääsee omaan säilytystilaan henkilökohtaisen linkin kautta. Säilytystilassa on tehtävään annetut tiedot, ja johon myös opiskelija voi vapaasti ladata omia tiedostoja (kuva 34).



Kuva 34. Opiskelijan säilytystilan näkymä

Klikkaamalla tehtävän otsaketta nähdään tehtävätilaan kirjautuneen opiskelijat sekä heidän tekemät säilytystilat. Tässä tilassa myös nähdään ovatko opiskelijat palauttaneet tehtäviä omiin säilytystiloihinsa.

Opiskelijoiden säilytystilat voi ladata ylhäällä olevasta ”Download repositories” valinnasta, jonka avulla osallistujien kurssivastaukset voi tallentaa omalle tietokoneelle. Osallistujien omia säilytystiloja ja tallennuksia voi tarkastella myös valitsemalla osallistujan kohdalla ”View repository”. Luokkatilan muodostajan näkymässä näkyy juuri luotu toimeksianto johon opiskelijat voivat palauttaa tehtäviä (kuva 35).



Kuva 35. Luotu tehtävä

5 Pohdinta

Ohjelmointiympäristön työkalujen asentamisessa joutuu opiskelija näkemään jonkin verran vaivaa. Sovellusten käyttö kuitenkaan ei todennäköisesti ole ongelma alan opiskelijoille, jotka mahdollisesti tulevat työssään käyttämään kyseisenlaisia työkaluja. Esimerkiksi versionhallinta GitHub on yleisesti ohjelmointialalla yrityksillä käytössä oleva työkalu, jonka käytön opettelu on hyödyllistä jo opiskeluaikana. Työkalut on suunniteltu käytettäväksi ohjelmointikursseille, esimerkiksi web-ohjelmointiin. Tällaisessa tilanteessa raportointityökalun rakentaminen osuu kurssin ohjelmoijaopiskelijan aihepiirin sisään.

5.1 Jatkokehitys

Kyseisten työkalujen käyttö ilmenee ongelmaksi tilanteessa, kun valmistaja lopettaa tuen sovellukselle tai palvelulle. Tähän opinnäytetyöhön valitut työkalut ja sovellukset ovat suurten yritysten tarjoamia yleisiä palveluita, jolloin niiden tuen loppuminen lähivuosina on epätodennäköistä. Teknologian kehittyessä jatkuvasti käytettyjen työtapojen ja työkalujen uudelleenarviointi on tärkeää.

Työkalujen tarpeellisuutta ohjelmointikurssin tavoitteisiin on myös hyvä arvioida erikseen kurssikohtaisesti. Joissain tilanteissa ohjelmointiajan seuraaminen ei ole hyödyllistä, jos varsinaista ohjelmointityötä ei ole kurssilla riittävästi. Tällöin ajan seurannan voi hoitaa jollakin muulla työkalulla, tai riippuen pedagogisista tarpeista sitä ei tarvitse seurata ollenkaan.

Jatkokehityksenä raportoinnin ja sitä varten luotavan sivusto olisi mahdollista lisätä palvelimelle, jossa se on saatavilla ja luettavissa internetin kautta. Jotkut koulut jakavat opiskelijoilleen palvelintilaa, jota voisi käyttää hyväksi lisäämällä raportointisivu internetpalvelimelle. Tämän suhteen kuitenkin herää uusia ongelmia, kuten tietoturvallisuus, kun mahdollisesti koulun ulkopuoliset henkilöt voivat päästä tietoihin käsiksi. Sisäverkossa sivun ylläpitäminen on kuitenkin usein turvallista, koska siihen pääsee ainoastaan paikallisesti käsiksi ja usein sisäverkkoon kirjaudutaan henkilökohtaisella tunnuksella.

Tätä opinnäytetyötä tehdessä teknisen toteutuksen ohjelmistoympäristöä ei tullut testattua opiskelijoilla, joka olisi ollut jatkokehityksen kannalta tärkeää. Joitakin palvelujen asentamiseen ja käyttöön liittyvä sudenkuoppia on hankala ennustaa, ennen kuin kokeilee niiden käyttöä käytännössä. Eri palvelujen yhteensopivuudessa voi mahdollisesti ilmetä ongelmia, vaikka palvelut olikin suunniteltu tätä silmällä pitäen.

5.2 Ammatillinen kasvu

Hyöty kurssin ohjaajalle ympäristön käytöstä tulee siinä, että opiskelijat itseohjautuvasti voivat tutustua kurssimateriaaliin, kuitenkin opettajan ohjaamalla tavalla. GitHub Classroom-palvelun avulla kurssin tehtävien lataaminen ohjaajan työasemalle on helppoa, ja näin palautettavien tehtävien hallinta on myös helppoa. Ohjaaja voi kommentoida jokaisen opiskelijan omaan säilytystilaan tehtävistä ja kysymyksistä, sekä voi suoraan palvelusta tarkastaa ohjelmointikoodin. Koska ohjaajan jakama materiaali voidaan kopioida opiskelijalle suoraan, on kurssin vuosittainen uusiminen todella helppoa.

Ohjaajan täytyy ainoastaan perustaa uusi luokkatila, ja jakaa uusi linkki opiskelijoille.

Opinnäytetyön aikana tuli huomattua, mitkä tekijät vaikuttavat eniten ohjelmointiopiskelussa menestymiseen, ja näitä ajatuksia ja tekniikoita on helppo käyttää hyväksi henkilökohtaisessa opiskelussa. Työkalujen vertailu ja niihin tutustuminen taas toi kokemusta erilaisten ohjelmointityössä tarvittavien välineiden käytöstä, esimerkiksi ohjelmistosäilön käyttäminen GitHub palvelussa tuli työtä tehdessä tutuksi.

Lähteet

Alvaris Falcon. 2017. Top 10 Websites to Learn Coding (Interactively) Online. HKDC. https://www.hongkiat.com/blog/sites-to-learn-coding-online/. 14.10.2018

Anabela Gomes & António José Mendes. 2007. An environment to improve programming education. University of Coimbra. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5458&rep=rep1&type=pdf. 13.5.2019.

Anders Berglund & Anna Eckerdal. 2015. Learning Practice and Theory in Programming Education. IEEE. http://uu.diva-portal.org/smash/get/diva2:811311/FULLTEXT01.pdf. 12.12.2018.

Bureau of Labor Statistics, U.S. Department of Labor. 2018. Occupational Outlook Handbook, Software Developers. Bureau of Labor Statistics, U.S. Depart-ment of Labor. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm. 11.4.2019.

Dianna L. Van Blerkom. 2008. College Study Skills: Becoming a Strategic Learner. Wadsworth 14.10.2019.

Edx. 2019. CS50's Introduction to Computer Science Harvard University. edX Inc. https://www.edx.org/course/cs50s-introduction-computer-science-harvardx-cs50x. 2.5.2019.

Evans Data Corporation. 2017. Global Developer Population and Demographic Study 2017 Vol. 1. Evans Data Corporation. https://evansdata.com/reports/viewRelease.php?reportID=9. 10.3.2019.

GitHub Classroom. 2018. GitHub. https://classroom.github.com/. 14.11.2018.

Hanna Salovaara. 2004. Oppimisen teoriasta tukea tieto- ja viestintätekniikan pedagogiseen käyttöön. Oulun Yliopisto. http://tievie.oulu.fi/verkkopedagogiikka/index.html 10.4.2019.

Henrik Muukkonen. 2017. Koodaripula iski Suomeen - palkat jopa 15 000 euroa kuussa. Talouselämä. https://www.talouselama.fi/uutiset/koodaripula-iski-suomeen-palkat-jopa-15-000-euroa-kuussa/013efa5f-f25b-37c8-bed8-04054362a5f6. 11.5.2019.

Hugo. 2018. GoHugo. https://gohugo.io/. 14.11.2018.

Jens Bennedsen. 2008. Teaching and Learning Introductory Programming. University of Oslo. https://www.duo.uio.no/bitstream/handle/10852/9962/bennedsen.pdf?sequence=1.

13.5.2019.
Karelia-ammattikorkeakoulu. 2018. Tradenomi, tietojenkäsittely. Karelia-ammattikorkeakoulu.
http://www.karelia.fi/fi/koulutus/amk-tutkinnot/tradenomi-tietojenkasittely.
12.5.2019.
Kielitoimiston sanakirja 2018. Kotimaisten kielten keskus.
https://www.kielitoimistonsanakirja.fi/.
20.5.2019.
Leo Kosola. 2015. Kymmenet pääsevät verkkokurssilla sisään Helsingin Yliopistoon. Yleisradio.
https://yle.fi/aihe/artikkeli/2015/11/04/kymmenet-paasevat-verkkokurssilta-sisaan-helsingin-yliopistoon.
13.5.2019.
Mooc.fi. 2018. Helsingin yliopisto, tietojenkäsittelytieteen laitos
http://mooc.fi/courses/2018/ohjelmoinnin-mooc/.
13.5.2019.
Opetushallitus. 2014, Tiedote uudesta opetussuunnitelmasta. Opetushallitus.
http://www.oph.fi/ajankohtaista/tiedotteet/101/0/uudet_opetussuunnitelmien_perusteet_paatetty.
11.5.2019.
Pekka Neittaanmäki, Päivi Kinnunen. 2016. TYÖTTÖMYYS IT-ALALLA KOKO SUOMESSA JA MAAKUNNISSA 2006–2015. Jyväskylän yliopisto.
10.3.2019.
Päivi Kinnunen. 2009. CHALLENGES OF TEACHING AND STUDYING PROGRAMMING AT A UNIVERSITY OF TECHNOLOGY VIEWPOINTS OF STUDENTS, TEACHERS AND THE UNIVERSITY. Aalto yliopisto.
Slack. 2019. Slack Technologies, Inc.
https://slack.com/.
14.11.2019.
Susan Bergin & Ronan Reilly. 2005. The influence of motivation and comfort-level on learning to program. Psychology of Programming Interest Group.
http://eprints.maynoothuniversity.ie/8685/.
14.11.2019.
Tony Jenkins. 2001. The Motivation of Students of Programming. University of Leeds.
http://bioinfo.uib.es/~joe/semdoc/p53-jenkins.pdf.
20.5.2019.
Visual Studio Code. 2019. Microsoft
https://code.visualstudio.com/.
1.11.2019.
Wakatime. 2019. Wakatime
https://wakatime.com/.
1.11.2019.